PROYECTO DE FIN DE CARRERA

OPTIMIZACIÓN MULTIOBJETIVO PARA CLASIFICACIÓN DE DATOS DESBALANCEADOS

Autor: Grande Benito, Pablo Lázaro

Tutores: Aler Mur, Ricardo
Galván León, Inés Mª

Octubre, 2010

ÍNDICE

1.- INTRODUCCIÓN

Desde los inicios de la informática, el ser humano se ha interesado por dotar a las máquinas creadas por él de las capacidades de una persona. El gran reto reside en confeccionar un artilugio que se comporte y reaccione del mismo modo que el ser humano, y para ello se le debe de conceder una inteligencia propia.

Dentro de las ciencias de la computación, la rama de **inteligencia artificial** (IA) se encarga del desarrollo de hardware y software que tenga comportamientos inteligentes. Para ello, se estudia en gran medida el razonamiento y aprendizaje de las personas y animales, y qué mecanismos/técnicas utilizan para resolver los distintos problemas y situaciones a las que son expuestos. La parte del cuerpo humano que más fascina, y por tanto se ha realizado una gran dedicación a su estudio, es el cerebro. Excepto en las tareas basadas en el cálculo aritmético simple, actualmente el cerebro humano es superior a cualquier computador: reconocimiento de imágenes, interpretación de sonidos, etc., en general en tareas de percepción. Ningún algoritmo conocido es capaz de emular de manera flexible estas funciones.

Las **redes de neuronas artificiales** (RNAs) surgen como un intento de desarrollar sistemas que emulen las características del cerebro, para conseguir su sofisticada capacidad de procesamiento de información. Han sido diseñadas como modelos extremadamente simplificados del funcionamiento del cerebro, y aunque sea muy esquemático, este modelo presenta una riqueza sorprendente de estados y de comportamientos que ha sentado las bases de un modelo de memoria y aprendizaje como un fenómeno emergente colectivo: el sistema global presenta propiedades complejas que no pueden predecirse a partir del estudio individual de sus componentes.

La pretensión de las RNAs es sintetizar un sistema que realice la estructura neuronal del cerebro y desarrolle un equivalente algorítmico de los procesos de reconocimiento y aprendizaje (imitación de las capacidades del cerebro). Está basada en la interconexión de multitud de elementos de procesamiento (las neuronas), cada uno de los cuales presenta un comportamiento completamente local.

Hay una diferencia sustancial entre los modos de desarrollo de una red neuronal y una aplicación software convencional: la red no se programa, sino que se entrena. Se denomina aprendizaje o entrenamiento de la red al ajuste de los pesos sinápticos (que determina el grado de conexión entre las neuronas de la red) a partir de un conjunto de ejemplos o patrones. Las RNAs se utilizan para problemas de clasificación, asociación, agrupamiento, optimización y predicción. En nuestro caso nos centraremos en las tareas de clasificación.

La **clasificación** es la atribución de una clase específica a un objeto. Esta atribución necesita un cierto grado de abstracción para poder extraer generalidades a partir de los ejemplos de los cuales se dispone. Para una máquina, la clasificación de rostros, de datos médicos o de formas, son tareas bastante difíciles, en tanto que para un humano son cuestiones cotidianas. Por ejemplo, en el caso de caracteres manuscritos, es difícil enunciar una descripción general que tenga en cuenta todas las variaciones particulares de cada carácter. Una técnica que puede ser utilizada para resolver este problema es el aprendizaje. Así, el criterio para decidir si una imagen corresponde a una letra **A**

consiste en comparar si esta imagen es lo suficientemente similar a otras *A*'s vistas anteriormente; con este enfoque, uno no calcula la clasificación de letras, sino que se aprende a partir de ejemplos. El aprendizaje está implícito en el quehacer humano y forma parte imprescindible de las actividades intelectuales; y se podría definir desde un punto de vista pragmático, como la adaptación de los parámetros de un sistema (artificial o natural) para obtener una respuesta deseada frente a un estímulo externo. Esta definición amplia del aprendizaje puede ser formalizada con el paradigma de aprendizaje supervisado.

En este proyecto se entrenarán RNAs con aprendizaje supervisado, en concreto el **perceptrón multicapa**, en el ámbito de la clasificación. El aprendizaje de la red es supervisado, esto es que por cada patrón (ejemplo) presentado a la red se conoce de antemano a que clase pertenece. La red evaluará el patrón y lo clasificará, y en base a si lo ha clasificado correctamente, la red se ajustará (aprenderá) para incluirlo en la clase correcta.

Para el entrenamiento (aprendizaje) de una RNA, es necesaria una gran cantidad de patrones del dominio que se desea comprender. El problema reside cuando el conjunto de patrones no se encuentra equilibrado para todas las clases, esto es que existen clases con una gran cantidad de patrones y otras que contienen muy pocos. En estos casos, los algoritmos de aprendizaje tienden a generalizar en las clases mayoritarias a costa de generalizar mal en las minoritarias. La idea es replicar aquellos patrones minoritarios y mal clasificados, para que aumente el porcentaje de aciertos para estas clases, pero con el compromiso de no empeorar en demasía la clasificación en el resto de clases.

Por tanto, nuestra intención es definir un conjunto de patrones de entrenamiento (en los que están replicados los más "conflictivos") de la RNA que maximicen los distintos porcentajes de aciertos de clasificación para cada una de las clases. A este tipo de problemas, en los que se busca una solución que satisfaga en la mayor medida a varios factores, se conocen como **problemas de optimización multiobjetivo**. De las distintas técnicas que existen para la resolución de este tipo de problemas, una de las más flexibles es el uso de **algoritmos evolutivos para la optimización multiobjetivo**. Los algoritmos evolutivos se trata también de otro procedimiento de la inteligencia artificial, que están basados en la teoría de la evolución biológica. Nuestro algoritmo evolutivo nos indicará cuantas veces se debe replicar cada patrón de entrenamiento con el objetivo de maximizar los porcentajes de acierto de todas las clases simultáneamente.

En resumen, el objetivo de este proyecto es mejorar la clasificación de patrones realizada con una red neuronal artificial, especialmente para problemas desbalanceados, desde un enfoque de optimización multiobjetivo mediante la utilización de algoritmos evolutivos.

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo:

- **Capítulo 1. Introducción.** Se trata de la sección actual, en la que se hace una breve introducción al proyecto, se describe el propósito del mismo y se detalla el contenido de esta memoria.
- **Capítulo 2. Contexto de Trabajo.** En esta sección se describe el ámbito en el que se ha desarrollado el proyecto, detallando las técnicas y herramientas utilizadas. Se divide en dos apartados, en los que se explica el Perceptrón Multicapa y los Algoritmos Evolutivos para la Optimización Multiobjetivo.
- **Capítulo 3. Descripción del Sistema.** En este apartado se detalla la implementación del sistema y la funcionalidad que ofrece. En primer lugar se describe de manera formal el propósito del sistema. Seguidamente se definen los elementos que intervienen en el algoritmo evolutivo para la aplicación diseñada. Por último, se especifican dos variantes implementadas en base al algoritmo inicial.
- **Capítulo 4. Experimentación.** En este capítulo se muestran todos los experimentos realizados con el sistema desarrollado para cuatro dominios. Se desarrolla un sub-apartado por cada dominio, mostrando los resultados obtenidos para cada uno de los tres métodos (versiones) implementados, y realizando un análisis general de los resultados para dicho dominio.
- **Capítulo 5. Conclusiones y Trabajos Futuros.** En esta sección se muestran las conclusiones obtenidas del trabajo realizado. También se recogen las posibles líneas futuras de ampliación del presente proyecto.
- **Capítulo 6. Bibliografía.** Se muestran todos los documentos que se han empleado para la elaboración de este proyecto, así como aquellos que son de especial interés.

2.- CONTEXTO DE TRABAJO

En este capítulo se describirá el entorno en el que se ha desarrollado este proyecto. Para la elaboración del sistema, se han utilizado dos técnicas propias de la inteligencia artificial: las redes de neuronas artificiales y los algoritmos evolutivos. La primera de ellas se trata de un paradigma de aprendizaje inspirado en el funcionamiento del cerebro humano. La técnica de algoritmos evolutivos es un método de optimización y búsqueda de soluciones, basado en la teoría de la evolución.

En los siguientes apartados se expondrán con más detalle cada una de las dos técnicas utilizadas, describiéndose su propósito, funcionamiento, características, tipos, y el modo en el que se han utilizado en este proyecto. Estas técnicas no han sido implementadas directamente en el sistema, sino que se han utilizado dos herramientas que proporcionan la funcionalidad requerida, y han sido adaptadas para su utilización dentro del sistema. En el caso de las redes de neuronas se ha utilizado la librería FANN que implementa el perceptrón multicapa y el algortimo de retroproganación para el aprendizaje de la red, y para los algoritmos evolutivos una implementación del algoritmo NSGAII. Las características y funcionamiento de estas herramientas también serán detalladas en los siguientes apartados.

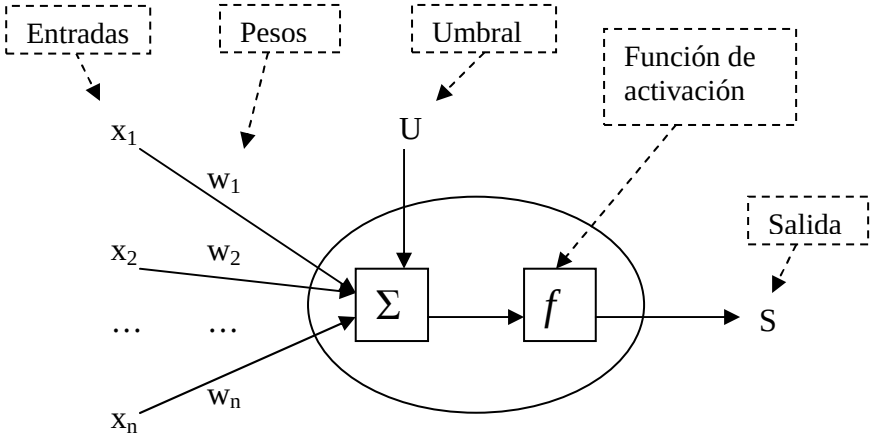
2.1.- Perceptrón multicapa

La red neuronal utilizada en este sistema es un perceptrón multicapa. Es un tipo específico de red neuronal, cuyas características principales es su aprendizaje supervisado y su arquitectura de múltiples capas de neuronas artificiales. En primer lugar se explicará el paradigma de red neuronal, seguidamente se describirá el caso específico del perceptrón multicapa, y por último la herramienta utilizada FANN.

2.1.1.- Red neuronal artificial

Durante varias décadas los científicos han buscado algoritmos capaces de procesar información al igual que el cerebro humano. En base al funcionamiento y estructura del cerebro animal, surgieron las redes de neuronas artificiales. La unidad básica de la comunicación nerviosa es la neurona. La neurona recibe la información de otras neuronas o receptores por medio de impulsos a través de las dendritas. La neurona procesa la información obtenida: si las señales recibidas superan un cierto umbral, producen una señal de activación. Dicha señal es transmitida mediante impulsos, a través del axón, a las neuronas subsiguientes o a las células efectoras.

En el modelo computacional, la unidad básica es la neurona artificial, cuyo diseño es similar a la neurona biológica. La neurona artificial recibe un conjunto de señales procedentes del mundo exterior o de otras neuronas. Estas señales de entrada se reciben a través de unas conexiones, las cuales tienen un peso asociado. La neurona artificial procesa la información recibida mediante la realización de operaciones simples, produciendo una señal de salida en base a las señales de entrada.



Neurona artificial

A partir de las señales de entrada (x_1, x_2, …., x_n), de los pesos (w_1, w_2, …., w_n) asociados a las conexiones de dichas entradas y del umbral (u) de la neurona artificial, se realiza el siguiente sumatorio:

$$NET = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n + u = \sum_{i=1}^{i=n} \left(x_i \cdot w_i \right) + u$$

Ecuación 1: Sumatorio de entradas, pesos y umbral de una neurona artificial

La salida de la neurona artificial será el resultado de aplicar la función de activación sobre el resultado obtenido en el sumatorio anterior, es decir: S = f(NET). Existen distintas funciones de activación, las más importantes son:

- Función lineal:

$$f(x) = x$$

- Función umbral:

$$f(x) = \begin{cases} f_1 & / x > 0 \\ -f_1 & / x \leq 0 \end{cases}$$

- Función gausiana:

$$f(x) = e^{\frac{-x^2}{2}}$$

- Funciones sigmoidales:
  - Función en (0, 1):

$$f(x) = \frac{1}{1+e^{-x}}$$

  - Función en (-1, 1):

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

El conjunto de neuronas artificiales conectadas entre sí, es conocido como red neuronal artificial. Las neuronas están conectadas entre ellas mediante una serie de arcos llamados conexiones, los cuales tienen un número real asociado, llamado **peso**. Las neuronas artificiales se distribuyen generalmente en capas de distintos niveles, con conexiones que unen las neuronas de las distintas capas y/o neuronas de una misma capa. Al modo en el que están conectadas y distribuidas las neuronas en una red neuronal, se le conoce como **arquitectura de la red**.

La parte más importante y que más caracteriza a una red de neuronas artificial es el aprendizaje de ésta. El aprendizaje de la red está basado en ejemplos, por lo que se necesita de un conjunto de ejemplos (conjunto de entrenamiento) que sea significativo y representativo del dominio a estudiar. El proceso de aprendizaje consiste en determinar los pesos de las conexiones de la red, de modo que se consiga unos valores de éstos que permitan resolver de manera eficiente el problema. Para ello, se introducen a la red de manera paulatina todos los ejemplos del conjunto de aprendizaje, y se van modificando los pesos en función de un esquema de aprendizaje. Esto se realiza de forma reiterada hasta alcanzar un criterio de convergencia: un número fijo de ciclos, hasta que el error obtenido sea menor a una cantidad determinada o cuando la modificación de los pesos sea irrelevante. En cuanto al esquema de aprendizaje, hay dos criterios:

- **Aprendizaje supervisado**: para cada patrón o ejemplo presentado a la red existe una respuesta deseada. La respuesta de la red se compara con su salida deseada, y en base a esa comparación se ajustan los pesos de la red. Este aprendizaje se utiliza en el perceptrón simple y multicapa, y en las redes de base radial.
- **Aprendizaje no supervisado**: no se especifica a la red cuál es la respuesta correcta. La red descubre las relaciones presentes en los ejemplos mediante reglas de aprendizaje. Las redes más importantes que utilizan este modelo de aprendizaje son: mapas autoorganizativos de Kohonen y el modelo ART.

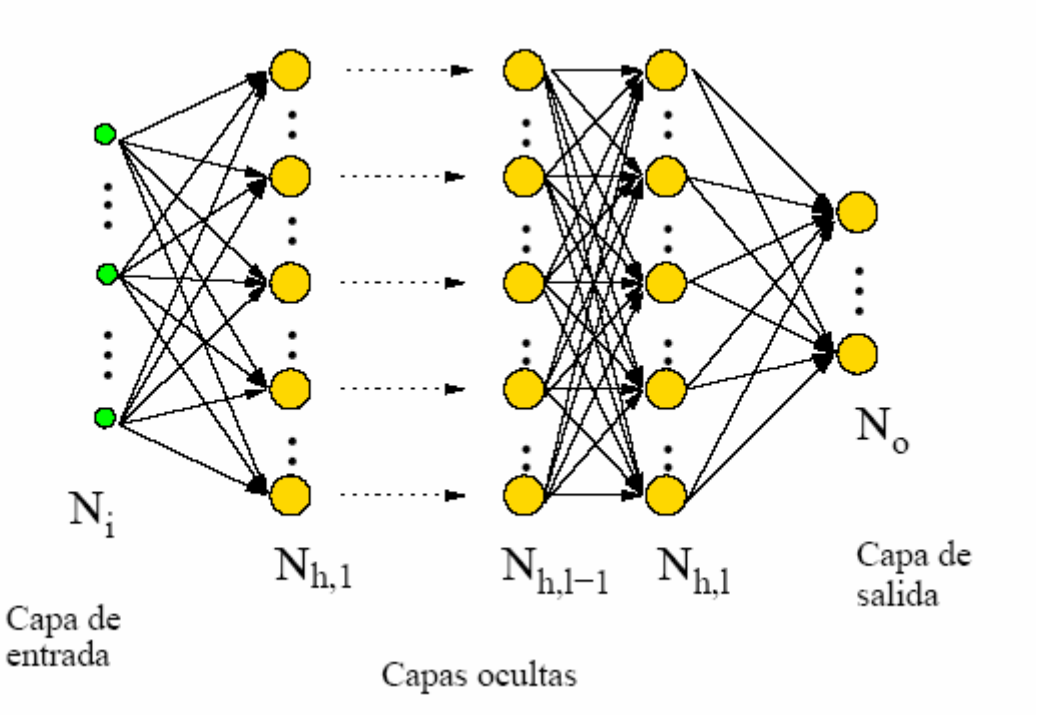
El entrenamiento o aprendizaje de la red con el conjunto de ejemplos puede llevar a un sobreajuste sobre dicho conjunto, reduciendo la capacidad de generalización de la red para nuevos casos. Por tanto, el conjunto de ejemplos es dividido en dos subconjuntos: el de **entrenamiento** y el de **validación** (test). El primero de ellos se utilizará para el ajuste de los valores de los pesos, y el segundo para medir la capacidad de generalización de la red neuronal. De este modo, para medir la eficacia de la red se utilizan datos que no han sido utilizados en su aprendizaje. Ambos conjuntos deben de ser independientes, y como se ha mencionado anteriormente, deben de ser significativos y representativos del problema a resolver.

2.1.2.- Perceptrón multicapa

El perceptrón multicapa es un modelo específico de red neuronal artificial, que fue propuesto por Rumelhart, Hinton y Williams en 1986 **[2]** para solventar las limitaciones del perceptrón simple, el cual no lograba resolver problemas que no fuesen linealmente separables. A continuación se describirán su arquitectura y su algoritmo de aprendizaje.

Las neuronas artificiales están agrupadas en capas, y cada una de ellas está conectada a todas las neuronas de la capa siguiente. Cada neurona procesa la información recibida y propaga la respuesta a través de la conexión con todas las neuronas de la capa siguiente. De este modo, se propagan los valores de las variables de entrada hacia adelante. Las capas están diferenciadas en tres tipos:

- **Capa de entrada**: recibe los patrones del exterior.
- **Capas ocultas**: procesan la información proporcionada por la capa anterior y producen una información de salida a la siguiente capa.
- **Capa de salida**: a partir de la información ofrecida de la última capa oculta, proporcionan la respuesta de la red para cada patrón de entrada.



Arquitectura del perceptrón multicapa

Para calcular la activación en una neurona se procederá siguiendo el procedimiento descrito a continuación. Sea un patrón o vector de entrada $X = (x_1, x_2, \ldots, x_p)$, y el vector de salida $Y = (y_1, y_2, \ldots, y_q)$. Sea un perceptrón multicapa con C capas, de las cuales C-2 serán ocultas, y con n_c neuronas en la capa c. Sea $W^c=(w^c_{ij})$ la matriz de pesos asociada a las conexiones de la capa c con la capa c+1, donde w^c_{ij} representa el peso de la conexión de la neurona i de la capa c con la neurona j de la capa c+1. Sea U^c el vector de umbrales de las neuronas de la capa c. Se denota a^c_i a la activación de la neurona i de la capa c. La **activación de una neurona**, según la capa en la que se encuentre, se calculará del siguiente modo:

- Neuronas de la capa de entrada (a^1_i):

$$a_i^1 = x_i \quad \text{para } i = 1, 2, \ldots, n_1$$

- Neuronas de las capas ocultas, a^c_i es la activación de la neurona i de la capa c, siendo c = 2, 3, …, C-1:

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} \cdot a_j^{c-1} + u_i^c \right) \text{ para } i = 1, 2, \ldots, n_c$$

- Neuronas de la capa de salida (a^C_i):

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} \cdot a_j^{C-1} + u_i^C \right) \text{ para } i = 1, 2, \ldots, n_C$$

La función de activación (f) generalmente es sigmoidal.

Como se puede observar, el número de neuronas en la capa de entrada y salida viene definido por el problema a resolver (n_1=p y n_{C-1}=q), dependiendo de la codificación de la información. Sin embargo, el número de capas ocultas y el número de neuronas que conforman cada una de estas capas, no viene establecido *a priori*. Estos valores se establecen mediante prueba y error.

El aprendizaje del perceptrón multicapa se realiza de manera supervisada, por lo que por cada patrón de entrada $X = (x_1, x_2, \ldots, x_p)$ a la red es necesario disponer de un patrón de salida deseada $S = (s_1, s_2, \ldots, s_q)$. El objetivo es encontrar los conjuntos de parámetros de la red: pesos W y umbrales U, tales que minimicen el error entre la salida dada por la red y la salida deseada. Por tanto, minimizar el error:

$$E = \frac{1}{N} \sum_{n=1}^{N} e(n) \quad ; \quad e(n) = \frac{1}{2} \sum_{i=1}^{n_C} (s_i(n) - y_i(n))^2$$

Donde N es el número de patrones y $e(n)$ es el error cometido por la red para el patrón n, siendo el vector de salidas de la red $Y(n) = (y_1(n),\ldots,y_p(n))$ y el de salidas deseadas $S(n) = (s_1(n),\ldots,s_q(n))$ para dicho patrón n.

Al utilizarse funciones de activación no lineales, hace que la respuesta de la red sea no lineal respecto a los pesos, por lo que nos encontramos con un problema de minimización no lineal. Por tanto, se realiza una adaptación de los parámetros siguiendo una dirección de búsqueda, que en el caso del perceptrón multicapa es la dirección negativa del gradiente de la función de error E (método del **descenso del gradiente**). El ajuste de los pesos se hace casi siempre por patrones (métodos de gradiente estocástico), que consiste en la sucesiva minimización de los errores para cada patrón $e(n)$, en lugar de minimizar el error global E. Aplicando este algoritmo, cada peso w se modifica para cada patrón de entrada n de acuerdo con la siguiente ley de aprendizaje:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w}$$

Esta variación del peso viene influenciada por el parámetro α, conocido como tasa o **razón de aprendizaje**. Este parámetro controla la magnitud del desplazamiento de los pesos en la superficie del error siguiendo la dirección negativa del gradiente. Por tanto, influye en la convergencia del algoritmo. A valores altos la convergencia es más rápida pero puede tener consecuencias negativas (saltarse un mínimo, oscilar alrededor de un mínimo, etc.); mientras que a valores pequeños conllevan una convergencia más lenta.

La aplicación del descenso del gradiente al perceptrón multicapa se conoce como algoritmo de **retropropagación** o ***regla delta generalizada***. Esta denominación se debe a que el error cometido en la salida de la red es propagado hacia atrás, transformándolo en un error para cada una de las neuronas ocultas de la red.

A continuación se muestran las ecuaciones finales tras desarrollar el método del descenso del gradiente en el caso del perceptrón multicapa. Es necesario diferenciar dos casos: uno para los pesos de la capa oculta C–1 a la capa de salida y para los umbrales de las neuronas de salida; y otro para el resto de los pesos y umbrales de la red.

En el caso de los pesos y umbrales de la última capa, se define el término δ asociado a la neurona i de la capa de salida (capa C) y al patrón n, $\delta_i^C(n)$, del siguiente modo:

$$\delta_i^C(n) = -(s_i(n) - y_i(n)) \cdot f'(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} \cdot a_j^{C-1} + u_i^C)$$

Siendo w_{ji}^{C-1} el peso de la conexión de la neurona j de la capa $C-1$ a la neurona i de la capa de salida, dicho peso se modificará siguiendo la siguiente ecuación:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha \cdot \delta_i^C(n) \cdot a_j^{C-1}(n)$$

para j=1,2,…,n_{C-1}, i=1,2,…,n_C

Generalizando dicha ecuación, se puede obtener la expresión que determina la modificación de los umbrales de la capa de salida:

$$u_i^C(n) = u_i^C(n-1) + \alpha \cdot \delta_i^C(n)$$

para i=1,2,…,n_C

Para los pesos de la capa c a la capa $c+1$ y umbrales de las neuronas de la capa $c+1$ para c=1,2,…,C-2. Se define el valor δ para la neurona j de la capa $c+1$, $\delta_j^{c+1}(n)$, como:

$$\delta_j^{c+1}(n) = f'(\sum_{k=1}^{n_c} w_{ki}^c \cdot a_k^c + u_j^c) \cdot \sum_{i=1}^{n_{c+1}} \delta_i^{c+2}(n) \cdot w_{ji}^c$$

Siendo w_{ki}^c el peso de la conexión de la neurona k de la capa c a la neurona i de la capa c+1, dicho peso se modificará siguiendo la siguiente ecuación:

$$w_{kj}^c(n) = w_{kj}^c(n-1) + \alpha \cdot \delta_j^{c+1}(n) \cdot a_j^c(n)$$

para k=1,2,…,n_c, j=1,2,…,n_{c+1}, c=1,2,…,C–2

Generalizando de nuevo, la ley de aprendizaje para el resto de los umbrales será:

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \alpha \cdot \delta_j^{c+1}(n)$$

para =1,2,…,n_{c+1}, c=1,2,…,C–2

Cada neurona de salida distribuye hacia atrás su error (valor δ) a todas las neuronas ocultas que se conectan a ella ponderado por el valor de la conexión. Así, cada neurona oculta recibe un cierto error (δ) de cada neurona de salida, siendo la suma el valor δ de la neurona oculta. Estos errores se van propagando hacia atrás, llegando a la primera capa.

Conociendo ahora la arquitectura, funcionamiento y aprendizaje del perceptrón multicapa, se describe a continuación los pasos del proceso de aprendizaje:

1. Se inicializan los pesos y umbrales de la red (valores aleatorios próximos a 0).
2. Se presenta un patrón n de entrenamiento, ($X(n)$,$S(n)$), y se propaga hacia la salida, obteniéndose la respuesta de la red $Y(n)$.
3. Se evalúa el error cuadrático, $e(n)$, cometido por la red para el patrón n.
4. Se aplica la regla delta generalizada para modificar los pesos y umbrales:
    a. Se calculan los valores δ para todas las neuronas de la capa de salida.
    b. Se calculan los valores δ para el resto de las neuronas de la red, empezando desde la última capa oculta y retropropagando dichos valores hacia la capa de entrada.
    c. Se modifican pesos y umbrales.
5. Se repiten los pasos 2, 3 y 4 para todos los patrones del conjunto de entrenamiento, completando así un ciclo de aprendizaje.
6. Se evalúa el error total E cometido por la red. Éste es el error de entrenamiento.
7. Se repiten los pasos 2, 3, 4, 5 y 6 hasta alcanzar un mínimo de error de entrenamiento, para lo cual se realizan m ciclos de aprendizaje. Puede utilizarse otro criterio de parada: pequeña variación del error de entrenamiento, aumento del error del conjunto de test, etc.

Para realizar el cálculo del conjunto de validación (o de test), se realizan únicamente los pasos 2 y 3 con una red de neuronas ya entrenada (o en proceso de entrenamiento), calculando posteriormente el error total E cometido por la red para todos los patrones del conjunto de test. Este error es conocido como error de validación o de test.

2.1.3.- Librería FANN

Para la utilización y manejo del perceptrón multicapa utilizado en el proyecto, se han usado las herramientas y funcionalidades que proporciona la librería FANN. *Fast Artificial Neural Nerwork* (FANN) es una librería de código abierto de redes de neuronas, que implementa redes neuronales multicapa en código C. Es soportado por cualquier plataforma Windows, Linux y Mac OS X, permitiendo su utilización mediante el uso de PHP, C++, .Net, Ada, Python, Delphi, etc.

En este proyecto se ha utilizado la versión 2.0.0, y para integrarla dentro del sistema se ha utilizado directamente el código fuente C sin utilizar las interfaces gráficas proporcionadas.

Para más información, se puede acceder a su página web, en donde se encuentra la documentación y en la que también se puede descargar la librería al completo: http://leenissen.dk/fann/index.php

La librería ofrece diversas funciones en C para crear, parametrizar, entrenar, etc. redes de neuronas. En primer lugar, los datos deben contenerse en un fichero que los contenga con el siguiente formato:

- En la primera línea se debe de indicar, en el siguiente orden: número de patrones, número de entradas y número de salidas; separados por un espacio.
- Por cada patrón, se escribirán dos líneas. En la primera de ellas se indicarán los valores de entrada del patrón y en la segunda los valores deseados en la salida para dicho patrón. Todos los valores estarán separados entre ellos por espacios.

Los valores que manejan las redes neuronales son de tipo real, por lo que para cualquier dominio que posea valores nominales, estos deberán ser transformados a valores reales. Además, como el proyecto se realiza en el ámbito de la clasificación, cada patrón estará incluido en una clase, por lo que los patrones tendrán tantas salidas como clases existan en el dominio. Según a la clase a la que pertenezca el patrón, la salida correspondiente a dicha clase tendrá el valor de 1, mientras que el resto de salidas valdrán 0.

Para la creación de una red de neuronas utilizaremos tres funciones proporcionadas por FANN:

- *fann_create_standard*. Con ella se crea una red de neuronas, y para ello es necesario indicar: el número de capas y por cada capa el número de neuronas que la conforman. En nuestro caso se utilizarán tres capas: entrada, oculta y salida. También los distintos pesos serán inicializados mediante esta función.
- *fann_set_learning_rate*. Con esta función se indica la razón de aprendizaje para una red de neuronas ya existente.
- *fann_set_activation_function_hidden*. Permite indicar la función de activación que tendrán todas las capas ocultas. Se ha seleccionado una función sigmoidal en el intervalo (0, 1): *FANN_SIGMOID*.

Una vez tenemos una red de neuronas creada, se guarda en un fichero mediante la función *fann_save*, para más adelante poder generar una red de neuronas exactamente igual a la definida con anterioridad. Para crear una red de neuronas a partir de una ya guardada en un fichero, se utiliza la función *fann_create_from_file*.

Para el entrenamiento de la red se pueden utilizar cualquiera de las siguientes funciones: *fann_train_on_data* o *fann_train_on_file*. Con ellas se entrenarán con los datos almacenados previamente en memoria o a partir de los datos de un fichero. A estas funciones se les deberá indicar el número de iteraciones a realizar, cada cuantas iteraciones se muestra un informe y el error deseado de la red.

Para evaluar un patrón mediante una red de neuronas FANN, se utiliza la función *fann_run*. A esta función se le pasará como parámetros la red de neuronas y el conjunto de entradas del patrón a evaluar. La función devolverá las salidas para dicho patrón que proporciona la red. Con esto, podemos realizar la función para cada patrón y después comparar la salida dada con la salida deseada.

2.2.- Algoritmos evolutivos para la optimización multiobjetivo

Este apartado se dedica a revisar conceptos relacionados con otras de las técnicas utilizadas en este proyecto, que se enmarca dentro de los algoritmos evolutivos multiobjetivo. A continuación se describirá el funcionamiento de los algoritmos evolutivos, qué es la optimización multiobjetivo y cómo abordarla mediante algoritmos evolutivos; y por último el funcionamiento y utilización de la herramienta utilizada NSGA-II.

2.2.1.- Algoritmos evolutivos

El término de algoritmos evolutivos se utiliza para englobar las distintas técnicas basadas en la evolución biológica. Se pueden diferenciar tres paradigmas principales:

- Algoritmos genéticos
- Programación evolutiva
- Estrategias evolutivas

Las técnicas utilizadas en este proyecto (en concreto el algoritmo NSGA-II) se basan en los algoritmos genéticos, por lo que nos centraremos en este tipo de procedimientos.

Los **Algoritmos Genéticos** (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde a los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin **[3]**. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas. Los principios de los Algoritmos Genéticos fueron establecidos por John Holland en 1975 **[4]** y se encuentran bien descritos en varios textos: Goldberg **[5]**, Davis **[6]**, Michalewicz, Reeves, etc.

Los AGs usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna una "puntuación" relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos que comparten algunas de las características de sus padres. De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior, conteniendo una mayor proporción de buenas características. Así, a lo largo de las generaciones las buenas características se propagan a través de la población. Si el AG ha sido bien diseñado, la población convergerá hacia una solución óptima del problema, o al menos hacia una solución cercana a la óptima.

Los individuos, que son posibles soluciones al problema, se representan como un conjunto de parámetros para dicho problema. Estos parámetros son conocidos como ***genes***. El conjunto de genes del individuo es nombrado como ***cromosoma***. Comúnmente esta cadena de variables es binaria, pero también puede ser de valores enteros o reales (en cuyo caso estaríamos ante una estrategia evolutiva). La codificación de los parámetros del problema en genes, se denomina ***genotipo***, mientras que el ***fenotipo*** es la decodificación del genotipo, con el fin de obtener los valores de los parámetros usados como entrada en la función objetivo del problema que se trata.

Por tanto, un individuo está formado por un cromosoma, el cual codifica una serie de parámetros para solucionar el problema. El conjunto de individuos que hay, se denomina población. Cada uno de los individuos es valorado mediante una función de evaluación, denominada ***fitness***, y que nos dará un valor que indica el nivel de bondad del individuo, es decir como de buena es la solución al problema dada por dicho individuo.

Los dos aspectos más importantes a la hora de diseñar un Algoritmo Genético son la codificación del cromosoma y la determinación de la función objetivo (función *fitness*).

En cada una de las generaciones se formará una nueva población a partir de la población inicial. Esto se realiza mediante el uso de los operadores evolutivos. Los principales son:

- **Selección**: Se trata de la simulación computacional de la Selección Natural postulada por Darwin. Mediante este operador se seleccionarán aquellos individuos (a partir de sus *fitness*) que producirán descendencia mediante la reproducción y la mutación. Tipos:
  - Ruleta
  - Jerárquica
  - Torneos

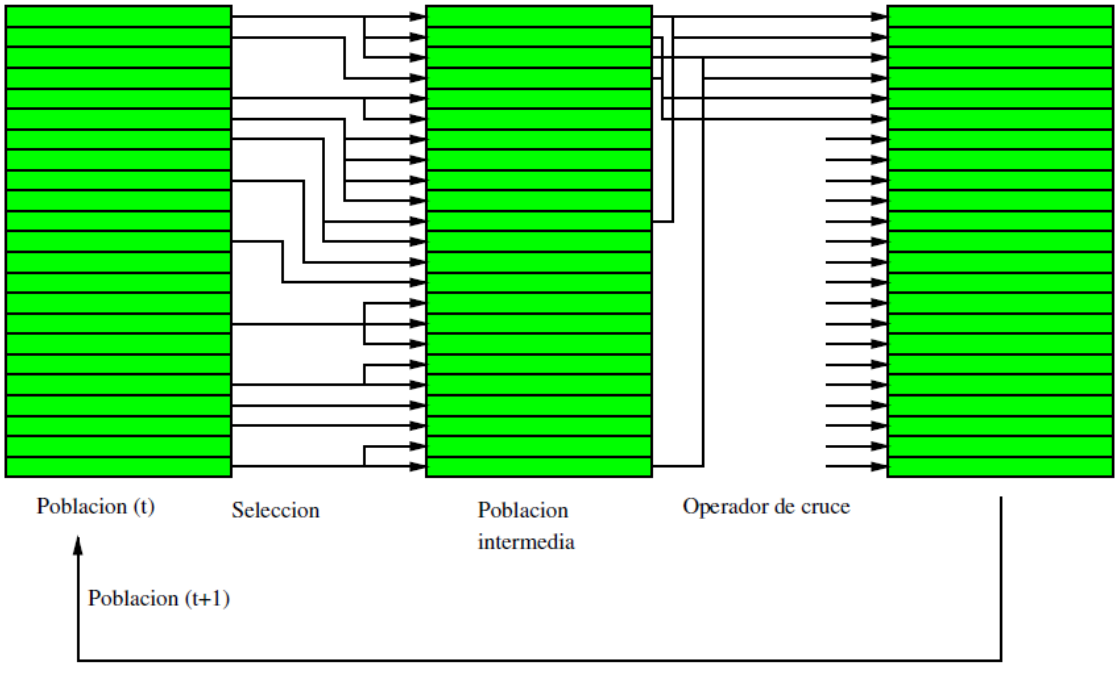
- **Reproducción**: Es la simulación de la reproducción sexual de los organismos. También es conocido como *cruce*, y consiste en generar uno o más individuos a partir de los individuos elegidos mediante la selección, mediante el intercambio de segmentos de sus cromosomas. Tipos:
  - Simple
  - De dos puntos
  - Uniforme

- **Mutación**: Es la simulación de las mutaciones que se producen al crearse un nuevo organismo. Consiste en realizar pequeñas modificaciones al cromosoma del individuo generado en la reproducción.

A la hora de conformar un AG, es necesario seleccionar los tipos de operadores genéticos a utilizar, así como las tasas de probabilidad de dichos operadores.

Una vez se han especificado todos estos aspectos se procede a realizar el algoritmo genético a partir de una población inicial. El Algoritmo Genético Canónico desarrollado por Holland realiza la siguiente secuencia:

1. Generación de una población inicial, que suele generarse mediante la creación de individuos con valores aleatorios.
2. Evaluación mediante la función objetivo (*fitness*) de cada uno de los individuos de la población.
3. Elección de dos individuos de la población mediante el operador de selección. La probabilidad que un individuo sea seleccionado es proporcional al *fitness* de dicho individuo.
4. Cruce, con cierta probabilidad, de los dos individuos seleccionados mediante el operador de reproducción elegido, creándose dos nuevos individuos.
5. Mutación de los dos nuevos individuos en base a una tasa de mutación.
6. Inserción de los dos individuos en la población de la nueva generación.
7. Realizar los pasos del 3 al 6 hasta que se complete la nueva población.
8. Ir al paso 2, y seguir la secuencia de pasos hasta que la población converja o se cumpla alguna condición de parada (por ejemplo un límite de número de generaciones).



Funcionamiento de un Algoritmo Evolutivo

El Algoritmo Genético se trata de un procedimiento heurístico, que es dependiente de los parámetros iniciales (tamaño de la población, operadores elegidos, tasas de probabilidad, etc.), por lo que es necesario ajustar lo mejor posible dichos parámetros para obtener buenas soluciones. Este ajuste se realiza teniendo en cuenta las características del problema, y mediante prueba y error.

2.2.2.- Problema de optimización multiobjetivo

Un problema multiobjetivo es aquel en el que se debe encontrar una solución común que satisfaga en la mayor medida a distintos factores. Al representar estos factores como funciones matemáticas, se pretende encontrar una solución que optimice todas las funciones de manera simultánea, definiéndose entonces un **problema de optimización multiobjetivo** (*Multiobjective Optimization Problem* – MOP). Estas funciones objetivo suelen estar en conflicto entre sí, lo que supone que una solución que optimice una de las funciones, causa valores peores en el resto de funciones. Por lo tanto, se pretende encontrar las soluciones que darían valores aceptables para todas las funciones objetivo. Sin embargo, a fin de simplificar su solución, muchos de estos problemas tienden a modelarse como mono-objetivos usando sólo una de las funciones originales y manejando las adicionales como restricciones.

Un problema de optimización multiobjetivo puede definirse formalmente como **[7]**:

Un MOP general optimiza una función

$$\mathbf{y} = \mathrm{F}(\mathrm{x}) = (f_1(\mathrm{x}), \ldots, f_\mathrm{k}(\mathrm{x})) \qquad (1)$$

sujeto a

$$\mathbf{g}(\mathrm{x}) = (g_1(\mathrm{x}), \ldots, g_m(\mathrm{x})) \leq 0 \qquad (2)$$

donde

$$\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X} \subseteq \mathbb{R}^\mathrm{n} \;;\; \mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y} \subseteq \mathbb{R}^\mathrm{k}$$

x es una variable de decisión vectorial *n*-dimensional, **y** es un vector objetivo *k*-dimensional, $\mathcal{X} \subseteq \mathbb{R}^\mathrm{n}$ denota el espacio de decisión, e $\mathcal{Y} \subseteq \mathbb{R}^\mathrm{k}$ denota el espacio objetivo. El conjunto de restricciones dadas por la ecuación 2 define la región de factibilidad $\mathcal{X}_f \subseteq \mathbb{R}^\mathrm{n}$ y cualquier punto **x** contenido en dicha región, se trata de una solución factible.

La resolución de un problema de optimización multiobjetivo puede plantearse como la búsqueda de un vector que representa el conjunto de variables de decisión y el cual optimiza (maximiza o minimiza) en conjunto de las funciones objetivo. En la mayor parte de los casos dichas funciones pueden estar en conflicto unas con otras.

Existen diversos métodos de solución para tratar el conflicto entre objetivos. El más conocido y en el que nos centraremos es la resolución mediante el concepto de eficiencia de Pareto. A continuación se definen los conceptos de dominancia y optimización de Pareto, aplicados a un problema de minimización:

- **Dominancia de Pareto**.
  Sean los vectores objetivo $\boldsymbol{u} = (u_1, \ldots, u_\mathrm{k})$ y $\boldsymbol{v} = (v_1, \ldots, v_\mathrm{k})$, se dice que **u** domina a **v**, si y sólo si:

$$\forall i \in \{1, \ldots, k\}, u_i \leq v_i \quad \wedge \; \exists i_0 \in \{1, \ldots, k\} \mid u_{i_0} < v_{i_0}$$

Es decir, **u** domina a **v**, si **u** es mejor (menor, en el caso de minimización) o igual a **v** en todos los objetivos y estrictamente mejor (menor) en al menos un objetivo. Si ninguno de los vectores domina al otro se dice que son indiferentes entre sí ya que ninguno puede ser considerado mejor que el otro considerando todos los objetivos.

• **Óptimo de Pareto**.

Se dice que una solución $\mathbf{x} \in \mathcal{X}_f$ es un optimo de Pareto con respecto a un conjunto $\Omega \subseteq \mathcal{X}_f$ si y sólo si:

$$\neg\exists \mathbf{x}' \in \Omega \mid \mathbf{v}' = \mathbf{F}(\mathbf{x}') \text{ domina a } \mathbf{u} = \mathbf{F}(\mathbf{x})$$

Por tanto, un vector **x** es un óptimo de Pareto si no existe otro vector de decisión factible $\mathbf{x}' \in \mathcal{X}_f$ que lo domine. En general, la solución en el sentido de Pareto al problema de optimización multiobjetivo no será única: la solución estará formada por el conjunto de todos los vectores no dominados, a los que se les conoce con el nombre de **frente de Pareto**.

• **Conjunto de Óptimos de Pareto**.

Dado un problema de optimización multiobjetivo **F**, el conjunto de óptimos de Pareto, denotado por $\mathcal{P}^*$, se define como:

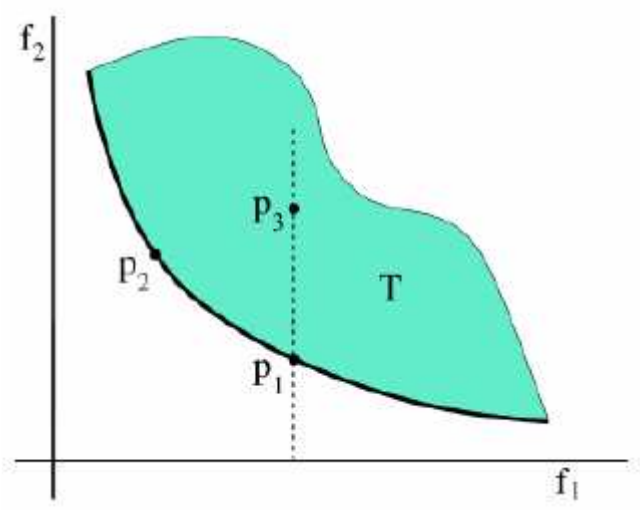
$$\mathcal{P}^* := \{\mathbf{x} \in \mathcal{X}_f \mid \neg\exists \mathbf{x}' \in \mathcal{X}_f \text{ t.q. } \mathbf{F}(\mathbf{x}') \text{ domina a } \mathbf{F}(\mathbf{x})\}$$

• **Frente de Pareto**.

Dado un problema de optmización multiobjetivo **F** y su conjunto de óptimos de Pareto $\mathcal{P}^*$, el frente de Pareto, denotado por $\mathcal{PF}^*$, se define como:

$$\mathcal{PF}^* := \{\mathbf{u} = \mathbf{F}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\}$$

es decir, $\mathcal{PF}^*$ es la imagen de $\mathcal{P}^*$ dada por **F**.



Frente de Pareto

La gráfica muestra el frente de Pareto (en trazo grueso) de un problema de optimización multiobjetivo con dos funciones objetivo: f_1 y f_2. Se puede observar como los puntos P_1 y P_2, pertenecientes al frente de Pareto, dominan al punto P_3, el cual obtiene el mismo valor para f_1 que P_1, pero el valor para f_2 es peor.

2.2.3.- Algoritmos evolutivos aplicados a la optimización multiobjetivo

El potencial de los algoritmos evolutivos para resolver problemas de optimización multiobjetivo se remonta a finales de los 1960s. Rosenberg en su tesis doctoral de 1967 **[8]** indicó la posibilidad de usar algoritmos genéticos en este dominio. Sin embargo, el primer intento real por extender un algoritmo evolutivo a problemas multiobjetivo es el *Vector Evaluated Genetic Algorithm* (VEGA) desarrollado por Schaffer en su tesis doctoral de 1984 **[9]** y presentado en la Primera Conferencia Internacional de Algoritmos Genéticos en 1985 **[10]**.

El gran potencial que proporcionan los algoritmos evolutivos para resolver problemas de optimización multiobjetivo, se debe a que se evoluciona una población de soluciones al problema. Con ello se consigue encontrar un conjunto óptimo de Pareto en una sola ejecución. Además, se aplican mecanismos que mantienen diversidad en la población (reproducción y mutación) para conseguir un conjunto de soluciones no dominadas lo más grande posible. Este tipo de algoritmos son conocidos como MOEAs (*Multi-Objective Evolutionary Algorithms*).

Se pueden considerar, en general, dos tipos principales de algoritmos evolutivos multiobjetivo:

- Los algoritmos que no incorporan el concepto de óptimo de Pareto en el mecanismo de selección del algoritmo evolutivo. Estos son, por ejemplo, los que usan funciones agregativas lineales. Ejemplos: VOW-GA y RW-GA.
- Los algoritmos que jerarquizan a la población de acuerdo a si un individuo es no dominado o no (usando el concepto de óptimo de Pareto). Ejemplos: MOGA, NSGA, NPGA, etc.

Asimismo, históricamente se puede considerar que han existido dos generaciones de algoritmos evolutivos multiobjetivo:

- **Primera Generación**: Caracterizada por el uso de jerarquización de Pareto y nichos. Son algoritmos relativamente simples, y que han comenzado a caer en desuso. Ejemplos: NSGA, NPGA, MOGA y VEGA.
- **Segunda Generación**: Se introduce el concepto de elitismo en dos formas principales: usando selección y usando una población secundaria. Estos algoritmos enfatizan la eficiencia computacional, pues se busca vencer la complejidad computacional de la jerarquización de Pareto y de las técnicas tradicionales de nichos. Desde finales de los 1990s los algoritmos evolutivos multiobjetivo que usan elitismo son vistos como el estado del arte en el área. Ejemplos: SPEA, SPEA2, NSGA-II, MOMGA, MOMGA-II, PAES, PESA, PESA II, etc.

En nuestro caso, el algoritmo que nos interesa es el NSGA-II, que será descrito en más detalle en el siguiente apartado.

2.2.4.- NSGA-II

El algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm*, versión 2) fue propuesto por K. Deb y sus estudiantes del Laboratorio de Algoritmos Genéticos del Instituto Tecnológico de Kanpur (India) en el año 2000 **[11]**. Surgió como una versión mejorada del algoritmo NSGA (año 1994), de quién heredó su estructura principal, pero incluyendo nuevas mejoras.

Las nuevas características que se añadieron solventaban distintos problemas detectados en la primera versión:

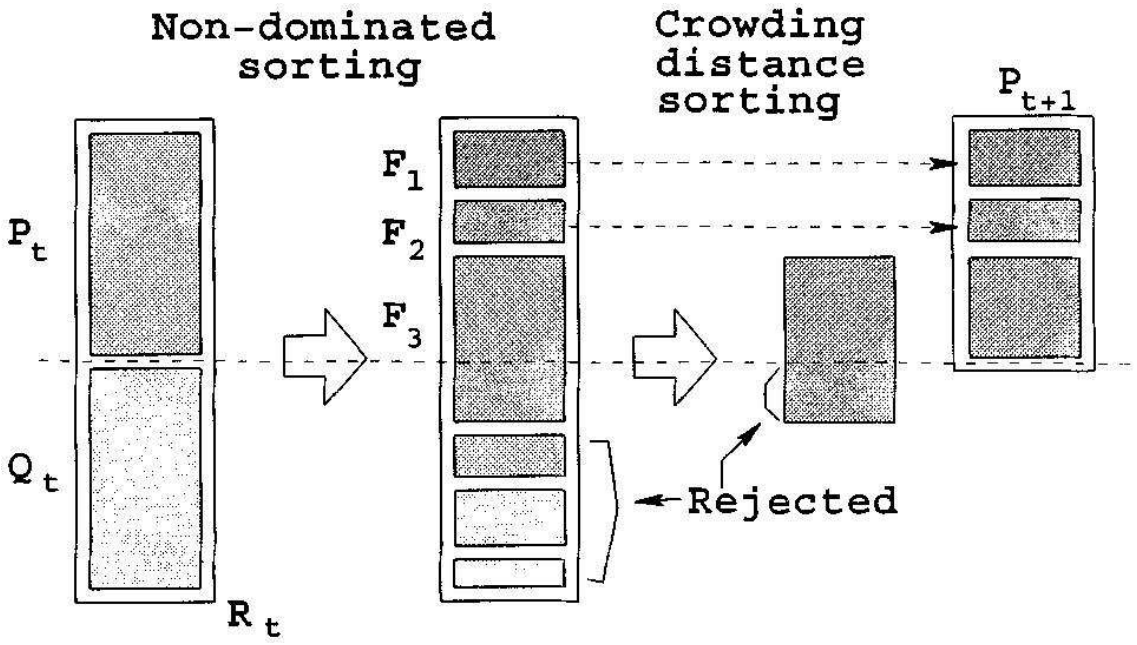
- Gran coste computacional en la ordenación de los individuos no dominados. Tenía una complejidad $O(mN^3)$, siendo m el número de funciones objetivo y N el tamaño de la población. En la nueva versión es reducido a $O(mN^2)$.
- Falta de elitismo. Como se ha comentado en el anterior apartado, la inclusión del *elitismo* mejora significativamente la funcionalidad del AG.
- Necesidad de indicar un parámetro para la técnica de *sharing*. En la nueva versión se añade una nueva técnica denominada *crowding*, que no requiere especificar parámetros adicionales para la preservación de diversidad en la población.

El algoritmo NSGA-II incorpora el cálculo de una distancia de *crowding*, como el operador utilizado para mantener la diversidad de la población, y así obtener un frente lo más disperso posible. La selección es realizada mediante torneo binario, utilizando como criterio de comparación el operador *crowded*. Según este criterio, el torneo lo gana el individuo con menor rango. Si el rango es el mismo, el torneo lo gana aquel individuo que tenga menor distancia de *crowding*.

A pesar de existir un gran número de MOEAs, como se puede observar en el anterior apartado, se ha optado por utilizar este algoritmo por varias razones. En primer lugar por tratarse de un algoritmo de la segunda generación, por lo que es más eficiente que los de primera generación al añadir el concepto de elitismo. Dentro de los distintos algoritmos que conforman esta segunda generación, los algoritmos más potentes y empleados son SPEA2 y NSGA-II, de entre los cuales se ha escogido este último por obtener muy buenos resultados **[7] [12]**, y porque ya ha sido utilizado en otros ámbitos, por lo que reduce el coste de aprendizaje de uno nuevo. Además, la implementación original en C realizada por los propios autores, permite una gran facilidad para acoplarlo con la otra herramienta utilizada (FANN), que también se encuentra implementada en este lenguaje.

A continuación, se describe el funcionamiento del algoritmo NSGA-II. En primer lugar se crea una población inicial P_0 de manera aleatoria, y es evaluada y ordenada en base a la no dominancia. Seguidamente se crea una población "hija" Q_0 de tamaño N a partir de P_0 mediante los operadores genéticos: selección mediante torneos, cruce y mutación. A partir de ahora, para $t \geq 1$, se sigue la siguiente secuencia, que corresponde con una generación del algoritmo genético:

1. $R_t = P_t \cup Q_t$ {combinación de padres e hijos}
2. F = ordenación_no_dominada(R_t) {$F = (F_1, F_2, \ldots)$, todos los frentes de R_t}
3. mientras $|P_{t+1}| + |F_i| \leq N$
   a. calcular_distacia_crowding(F_i)
   b. $P_{t+1} = P_{t+1} \cup F_i$
   c. $i = i + 1$
4. ordenación_operador_crowded(F_i)
5. $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ {coge los $(N - |P_{t+1}|)$ primeros elementos de F_i}
6. Q_{t+1} = crear_nueva_población(P_{t+1}) {con los operadores genéticos}
7. $t = t + 1$



Procedimiento del NSGA-II

Este proceso es repetido hasta que se cumpla la condición de parada (comúnmente es un número máximo de generaciones). La coste computacional de las operaciones utilizadas, en el peor de los casos, es la siguiente:

- Ordenación no dominada: $O(mN^2)$
- Asignación de distancias de *crowding*: $O(mN \log N)$
- Ordenación con el operador *crowded*: $O(2N \log(2N))$

Por tanto, la complejidad total del algoritmo es: $O(mN^2)$.

2.2.5.- Implementación de NSGA-II

En el proyecto se ha utilizado una implementación en C del algoritmo NSGA-II desarrollada por los mismos creadores del propio algoritmo: el Laboratorio de Algoritmos Genéticos de Kanpur (*Kanpur Genetic Algorithms Laboratory* – KanGAL). Este laboratorio dispone de una página web desde donde descargar de todo el *software* que han desarrollado: http://www.iitk.ac.in/kangal/codes.shtml. La versión que se ha utilizado es: *Multi-objective NSGA-II in code C: Original Implementation (for Windows and Linux).*

Para la utilización de este *software* es necesario modificar el código fuente del programa. En concreto, el fichero a modificar es "*fun-con.h*", que contiene la función de evaluación de los individuos. En él, se debe especificar qué valor asignar a cada una de las funciones objetivo, a partir de los valores del cromosoma del individuo. En este fichero también se pueden definir funciones de restricción, pero en nuestro caso no procede.

Para especificar los distintos parámetros del algoritmo, se solicitan a través de teclado al usuario antes de iniciar la ejecución del mismo. Para agilizar la ejecución del algoritmo, se ha modificado el fichero "*input.h*", la cual se encargaba de recoger los parámetros por teclado, y se recogen de un fichero que se le ha denominado "*config.txt*". En él se indicarán:

- Tamaño de la población.
- Número de generaciones.
- Probabilidad de cruce (entre 0 y 1). Fijado a 0'5.
- Probabilidad de mutación (entre 0 y 1). Fijado a 0'01.
- Índice de distribución para el cruce (entre 0 y 100). Fijado a 20.
- Índice de distribución para la mutación (entre 0 y 500). Fijado a 10.
- Límite inferior (inclusive) para todas las variables. Fijado a 1.
- Límite superior (exclusive) para todas las variables. Fijado a 6.
- Semilla para la generación aleatoria (entre 0 y 1). Fijado a 0'42.

La mayoría de los parámetros se han fijado a un valor, que es idéntico para todos los dominios, mientras que el tamaño de población y número de generaciones es diferente para cada una de las ejecuciones.

Existe una serie de parámetros que eran solicitados por teclado y que no se recogen en este fichero de configuración. Se tratan de la definición del cromosoma: longitud, tipo de valor (real, binario), etc.; y del número de funciones objetivo y de restricción del problema. Estos dependen del dominio, y se pueden definir una vez conocida la tipología del mismo, por lo que no es necesario indicarlos *a priori*. La definición de estos parámetros se expondrá en los próximos apartados.

3.- DESCRIPCIÓN DEL SISTEMA

3.1.- Introducción

El propósito de la aplicación desarrollada es la **optimización** de la **clasificación** realizada por una **red de neuronas** para un determinado dominio. La idea central es conseguir que la red mejore su aprendizaje de aquellos patrones difíciles de clasificar. Para ello, se **replicará** un determinado número de veces ciertos **patrones** del conjunto de entrenamiento de la red, con el objetivo de que la red de neuronas adquiera un mejor conocimiento del dominio. Los patrones a replicar se seleccionarán según los criterios que se explicarán más adelante.

El conjunto de entrenamiento (***T***) está constituido por una cierta cantidad de patrones del dominio, compuesto cada uno de ellos por unos atributos (x_n) y la clase (y_n) en la que se cataloga el patrón *n*. Por tanto, el conjunto de entrenamiento es definido como $T = (x_n, y_n)$, $n = 1 : |T|$, donde $|T|$ es el número de patrones de entrenamiento (la nomenclatura *"a : b"* está tomada de *MatLab* y se refiere a la lista de valores entre *a* y *b*). De este modo, el número de veces que se replicará cada patrón se determinará mediante el vector $z = (z_1, \ldots, z_{|T|})$. Al componente de *n* de *z* lo llamaremos z_n.

Para determinar el **número de veces** que se debe **replicar cada patrón** del conjunto de entrenamiento, es decir calcular ***z***, se ha utilizado un **algoritmo evolutivo**. Para este proyecto se ha decidido utilizar un algoritmo **multiobjetivo**. La razón es comprobar si intentando optimizar varios objetivos a la vez, se consigue explorar mejor el espacio de posibles redes y mejorar así su capacidad de generalización. A la optimización de objetivos secundarios con el fin de optimizar un objetivo primario (la capacidad de generalización), se la denomina multiobjetivización. Queda por definir qué objetivos se van a elegir para la optimización multiobjetivo. Parece razonable suponer que las redes que generalizan mejor se encontrarán entre aquellas que obtengan buenos porcentajes de acierto en todas las clases consideradas por separado. Así pues, si el problema de clasificación es de ***n* clases**, el planteamiento multiobjetivo consistirá en optimizar los *n* porcentajes de acierto correspondientes. Desde un punto de vista multiobjetivo, estos objetivos son apropiados, puesto que son objetivos contrapuestos: mejorar la precisión de una clase tiende a empeorar la de las demás.

Por otro lado, es sabido que los algoritmos de aprendizaje, incluidos las redes de neuronas, tienen dificultades para generalizar bien en problemas con una mala distribución de los datos en cada una de las clases (a estos problemas se los denomina *"imbalanced data sets"* o **datos desequilibrados**) **[15] [16]**. En estos problemas los algoritmos de aprendizaje tienden a generalizar bien en las clases mayoritarias (con más datos), a costa de generalizar mal en las minoritarias. Con el método propuesto en este proyecto, incluso aunque no se consiga mejorar el porcentaje de aciertos global, es posible que se encuentren redes que permitan mejorar la precisión en las clases minoritarias sin empeorar demasiado las mayoritarias. Ésta es otra de las hipótesis a comprobar en este proyecto.

Como optimizador multiobjetivo se ha utilizado el algoritmo **NSGA-II**. En suma, NSGA-II calculará un *z* que mejore la clasificación de la red de neuronas para cada una de las clases, de lo que se espera que optimice la clasificación en general para todo el dominio, aunque esto último es una hipótesis a comprobar en este proyecto.

Cada individuo de la población del algoritmo evolutivo representará una red de neuronas que ha sido entrenada en base al vector de patrones replicados *z*. Es decir, el conjunto de entrenamiento estará formado por los patrones originales (x_n,y_n), replicado cada uno de ellos tantas veces como indique z_n. Por tanto, tras la ejecución de NSGA-II se obtendrá un conjunto de redes de neuronas que optimizan la clasificación para cada una de las clases. En concreto, se obtendrá un **frente de Pareto** de **redes de neuronas**. De entre todas ellas se seleccionará la que mejor clasificación global posea.

Para el desarrollo de este sistema se han optado por varios criterios para: la selección de patrones a replicar, el conjunto de patrones para evaluar la red de neuronas y la selección del individuo (red de neuronas) del frente de Pareto obtenido por NSGA-II. De este modo, se han implementado **tres métodos distintos** según los criterios escogidos. A continuación se describirá cada uno de los métodos utilizados y se profundizará en los detalles del funcionamiento del algoritmo evolutivo en base al método descrito.

3.2.- Método 1

Este método es la aplicación **original** de la que se ha partido y del cual se han desarrollado posteriormente los otros dos métodos. Se fundamenta en **replicar** solamente aquellos patrones que han sido **mal clasificados** inicialmente por la red de neuronas. A continuación se detalla la implementación de los distintos elementos que intervienen en el algoritmo evolutivo: cromosoma, objetivos, función de *fitness* y selección de un punto del frente de Pareto.

3.2.1.- Cromosoma

Cada uno de los individuos de la población del algoritmo evolutivo representa una red de neuronas entrenada en base a *z*. El individuo está codificado por un cromosoma, que en este caso es el propio vector *z*.

Solamente se replicarán aquellos patrones que se han clasificado mal por la red de neuronas inicial. Para conocer cuáles son los mal clasificados, se entrena inicialmente una red de neuronas (RN_T) con el conjunto de entrenamiento original (T). Seguidamente se evalúa el mismo conjunto de patrones 'marcando' aquellos que la red no clasifica correctamente. El conjunto de patrones mal clasificados se denominará $M = \{ (x_n,y_n) \mid y_n \neq RN_T (x_n) \}$.

Por tanto, el vector *z* se reducirá sólo a aquellos patrones que no clasifica la red de neuronas inicial, resultando $z = (z_1,\ldots, z_{|M|})$. Este vector de valores es una lista ordenada de valores reales z_i comprendidos entre 1 y 5, ambos inclusive, que indica el número de veces que se replicará el patrón (x_i,y_i).

De este modo, el cromosoma estará compuesto por los z_i. A partir del cromosoma se creará una red de neuronas asociada. Para ello, se construye un nuevo conjunto de entrenamiento (T_z) que estará formado por los patrones que fueron bien clasificados ($T – M$) y por los patrones mal clasificados replicados tantas veces como se indique en *z* (M_z). Por tanto, el nuevo conjunto es: $T_z = T – M + M_z$. A partir de este conjunto se entrena una nueva red de neuronas, resultando RN_z. Esta red de neuronas será la que se utilizará para evaluar al individuo.

3.2.2.- Objetivos

Los **objetivos** del algoritmo evolutivo se corresponden con los **errores** que obtiene la red de neuronas de un individuo (RN_z) para cada una de las clases existentes. Se usa el error porque NSGA-II minimiza por omisión. Por tanto, habrá tantos objetivos como clases.

Definiremos T_c como el número de patrones de entrenamiento que pertenecen a la clase *c*, es decir aquellos tales que (x_i,c), $i = 1 : |T_c|$. De esta manera, el error que se produce para cada clase será igual al cociente del número de patrones de esta clase que han sido mal clasificados, entre el total de patrones de dicha clase. Esto viene dado por la siguiente ecuación (***Ecuación 1***):

$$E_c(z) = \frac{1}{|T_c|} \cdot \sum_{i=1}^{i=|T_c|} \delta(y_i, RN_z(x_i))$$

Ecuación 1: Error de clasificación de una clase para RN_z

La función $\delta(y_i,RN_z(x_i))$ tendrá el valor de 1 si el patrón (x_i,y_i) se clasifica equivocadamente con la red de neuronas RN_z, es decir $(x_i,y_i) \mid y_i \neq RN_z\ (x_i)$; y 0 en el caso de que sea clasificado correctamente.

3.2.3.- Función de *fitness*

El cometido de la función de *fitness* es evaluar cada uno de los cromosomas (*z*) de una población del algoritmo evolutivo. Esta evaluación del cromosoma es la base del algoritmo evolutivo para su proceso de búsqueda de la solución. El resultado de esta función es el valor de cada uno de los objetivos del algoritmo. Por tanto queda explicar la manera de cómo se obtienen dichos objetivos.

En cada generación, cada cromosoma *z* de la población actual es evaluado por la función de *fitness*, realizando el siguiente proceso:

1. Se construye el conjunto de patrones mal clasificados, replicando cada uno de ellos tantas veces como indique el cromosoma *z*. De este modo, obtendremos M_z.
2. Se construye el conjunto de entrenamiento para la red asociada al cromosoma *z*, es decir RN_z, mediante la unión de los patrones bien clasificados más los patrones mal clasificados replicados, por tanto: $T_z = T - M + M_z$.
3. La red de neuronas RN_z es entrenada con el conjunto T_z, mediante la utilización del algoritmo de retropropagación (*backpropagation*).
4. Se evalúa la red RN_z con el conjunto de patrones original (*T*), mediante la Ecuación 1, obteniéndose así el error de la red para cada una de las clases del dominio.
5. Cada uno de los valores obtenidos con la ecuación, es decir los errores de clasificación para cada clase, son suministrados al NSGA-II.

Hay que tener en cuenta diversos aspectos que se deben de realizar antes de comenzar el algoritmo NSGA-II. En primer lugar se entrenará, mediante *backpropagation*, una red de neuronas inicial (RN_T) con todo el conjunto de entrenamiento *T*. Seguidamente se evaluará la red con ese mismo conjunto, obteniéndose así el conjunto de patrones mal clasificados (*M*) y el de bien clasificados ($T - M$). Esta misma red, servirá para realizar una comparación con los resultados finales del sistema, y valorar así nuestro sistema. Por otra parte, se establecerán los pesos iniciales (ω_0) de las redes de neuronas que se generarán, tanto para RN_T como para las RN_z que se crearán durante el algoritmo evolutivo. De este modo todas las redes de neuronas generadas durante la ejecución de NSGA-II serán exactamente iguales inicialmente, por tanto se podrán comparar entre sí de forma imparcial.

3.2.4.- Selección de un punto del frente de Pareto

Tras ejecutarse el algoritmo evolutivo NSGA-II, éste proporciona la población de la última generación. Esta población está formada por los mejores individuos que se han generado durante el algoritmo. Cada uno de estos individuos está representado por un cromosoma, del cual se obtiene una red de neuronas: RN_z. Por tanto, el resultado son las redes de neuronas que han minimizado los objetivos. Recuérdese que los distintos objetivos que se han de conseguir es la minimización del error de clasificación de los patrones para cada una de las clases existentes en el dominio. A este conjunto de individuos que se ha obtenido se le conoce como frente de Pareto (véase apdo. 2.2.2).

Muchas de las redes clasificarán bien una de las clases mientras que el resto de clases las clasificarán peor. Los puntos del centro del frente clasificarán todas las clases de manera equilibrada. El propósito es la obtención de una red genérica que maximice el porcentaje de aciertos global. Para seleccionar la red de neuronas del frente de Pareto que más se acerca a nuestro objetivo, se evaluará cada una de ellas con el **conjunto de entrenamiento inicial** (*T*), obteniéndose el error global de clasificación (***Ecuación 2***):

$$E(z) = \frac{1}{|T|} \cdot \sum_{i=1}^{i=|T|} \delta(y_i, RN_z(x_i))$$

Ecuación 2: Error global de clasificación RN_z

Aquella red que obtenga el menor error global de clasificación, será la seleccionada como la mejor solución para este dominio, que la denominaremos RN_F. Aunque el conjunto de entrenamiento *T* haya sido utilizado durante el algoritmo evolutivo, se ha seleccionado este mismo conjunto de datos para la selección de la red pues se utiliza en ámbitos distintos: en el *fitness* se utiliza para la obtención del error de clasificación de una clase específica (E_c), mientras que ahora es utilizado para la adquirir el error de clasificación global (*E*).

Finalmente, la red RN_F será evaluada con el conjunto de test. Este conjunto de test es independiente del conjunto de entrenamiento (*T*), pues al inicio de la prueba se divide la totalidad de los datos del dominio en dos partes independientes: conjunto de entrenamiento y conjunto de test. Mediante la evaluación de la red inicial de la prueba (RN_T) y de la red final (RN_F) con el conjunto de test, se puede observar la mejora que se produce en la clasificación al evaluarse con un conjunto de datos independiente del algoritmo realizado.

3.3.- Variantes del método 1

En base al método que se ha descrito anteriormente, se han desarrollado otros dos métodos que utilizan criterios diferentes. Las diferencias radican en dos puntos concretos del sistema:

- Los patrones a replicar
- Conjunto de patrones que evaluará la red (*fitness*) y determinará el mejor del frente

3.3.1.- Método 2

La diferencia de este método con el método original radica únicamente en el segundo punto citado anteriormente. Mientras que en el método 1, se utiliza el conjunto de entrenamiento original (sin replicas de patrones) para la evaluación de la red y para la posterior elección del mejor punto del frente de Pareto, en este método se utiliza otro conjunto de patrones distinto.

El **conjunto de entrenamiento** original (*T*) es **dividido** al inicio de todo el proceso. Se divide en dos partes distintas: ***entrenamiento1*** *(T1)* y ***entrenamiento2*** *(T2)*. Estos dos conjuntos serán utilizados en situaciones diferentes.

El subconjunto entrenamiento *T1* se utilizará para el **entrenamiento** de las redes de neuronas, y por tanto determinará los cromosomas:

- Entrenamiento de la red inicial RN_{T1} para obtener los patrones mal y bien clasificados. A partir de los mal clasificados se formarán los cromosomas, que especificarán el número de veces a replicar dichos patrones.
- Para cada red neuronal asociada a un cromosoma (RN_z^{T1}), se entrenará con el conjunto *T1* con los patrones replicados según el cromosoma.

El subconjunto de entrenamiento *T2* se utilizará para la **evaluación** de la red de neuronas y elección del mejor punto del frente:

- En la función de *fitness*, en vez de evaluarse la red RN_z^{T1} con el conjunto *T*, será evaluada con el conjunto *T2*, por lo que la función objetivo será la siguiente (***Ecuación 3***):

$$E_c^{T1}(z) = \frac{1}{|T2_c|} \cdot \sum_{i=1}^{i=|T2_c|} \delta(y_i, RN_z^{T1}(x_i))$$

Ecuación 3: Error de clasificación en el método 2 para una clase para RN_z^{T1}

- Al finalizar el algoritmo evolutivo, para seleccionar el mejor punto del frente de Pareto se utilizará el conjunto de patrones *T2* en vez de *T*. Por tanto, la función para evaluar los puntos del frente será la representada en la ***Ecuación 4***:

$$E^{T1}(z) = \frac{1}{|T2|} \cdot \sum_{i=1}^{i=|T2|} \delta(y_i, RN_z^{T1}(x_i))$$

Ecuación 4: Error global de clasificación para RN_z^{T1} en el método 2

De este modo, se separa totalmente el entrenamiento de la red de neuronas de la evaluación de la misma, tanto para el cálculo de la función de *fitness* como para la elección del mejor punto del frente.

3.3.2.- Método 3

Este método es una variante del original en el criterio de selección de los **patrones a replicar**, es decir aquellos que "formarán" el cromosoma del algoritmo evolutivo. En el método 1 los patrones seleccionados son los que la red de neuronas RN_T clasifica erróneamente. En este método, el cromosoma estará construido a partir de los patrones que la red RN_T los clasifica difusamente o ambiguamente, a los que se denominará como **patrones difusos**. El resto del proceso se realizará exactamente igual. Para evaluar qué patrones son difusos se ha seguido un criterio, que se especificará a continuación.

En primer lugar se debe comprender como se han utilizado las redes de neuronas para determinar la clasificación de cada patrón. Las redes de neuronas empleadas tienen tantas salidas como clases existentes, relacionada cada salida con cada clase. La función de activación utilizada es sigmoidal, dando un valor entre 0 y 1. Aquella salida que obtenga el mayor valor, será la salida que se activa y por tanto el patrón será clasificado en la clase relacionada con tal salida.

Cuando los valores de las salidas de la red son muy próximos, se dice que su resultado es difuso, pues es difícil categorizar al patrón en una de las clases. Para seleccionar los patrones difusos, se calcula la diferencia entre las dos salidas de la red con mayor valor, si esta diferencia supera un **umbral** (se ha optado por 0'1), el patrón se considera difuso, en caso contrario no será difuso.

Con este método, se pretende obtener una red cuyos resultados sean más robustos, pues se aumenta la diferenciación entre las distintas clases, pues el algoritmo evolutivo se centrará en minimizar la ambigüedad en la catalogación de patrones. Es decir, en intentar clasificar correctamente los patrones ambiguos. Por tanto, se espera que se aumente su precisión y por consiguiente el porcentaje de aciertos de clasificación.

4.- EXPERIMENTACIÓN

En este capítulo se mostrarán todos los experimentos realizados con el sistema desarrollado en diversos dominios. Para cada uno de los dominios se expondrá la elección de la red más conveniente para éste, y los resultados obtenidos para cada uno de los tres métodos.

Los dominios que se han utilizado son cuatro: ***BUPA***, ***Car***, ***Thyroides*** y ***Balance-Scale***. Los datos han sido obtenidos del repositorio de datos para máquinas de aprendizaje, de la Universidad de California en Irvine (*UCI Machine Learning Repository*), disponibles en el enlace: http://archive.ics.uci.edu/ml/datasets.html

Para algunos de los dominios que se han escogido, se ha utilizado una validación cruzada. Estos son: *BUPA*, *Car* y *Balance-Scale*. En el caso del dominio *Thyroides* se ha utilizado un conjunto de entrenamiento y test ya diferenciados en el repositorio. A continuación se describe el proceso de validación cruzada.

Validación cruzada

Para solventar posibles sesgos en los conjuntos de entrenamiento y test seleccionados, y debido a que para ciertos dominios el conjunto de datos es bastante reducido, se ha utilizado **validación cruzada**. La validación cruzada (*cross validation)* consiste en dividir el conjunto de datos (*D*) en *n* partes iguales, que denominaremos D_i, y se realiza durante *n* veces lo siguiente:

- El conjunto de entrenamiento está formado por $D - D_i$ (i=1..n)
- El conjunto de test está formado por D_i (i=1..n)

Cada una de las *n* veces que se realiza esta división es conocida como ***fold***. Por cada *fold* se realizará una ejecución del algoritmo, utilizando los conjuntos de entrenamiento y de test creados según se ha descrito anteriormente. Por último, se realizará la media de los porcentajes de aciertos obtenidos en cada *fold*, permitiendo realizar una buena estimación de la eficacia del sistema.

La validación cruzada que se ha utilizado en este proyecto es de **5 *folds***. Cada uno de éstos se ejecuta independientemente, pudiéndose de este modo optimizar el rendimiento si se realizan todas las ejecuciones simultáneamente en una máquina con multiprocesador.

Parámetros

Los parámetros que se han seleccionado para el algoritmo NSGA-II, y que son idénticos para todos los dominios son:

- Número de generaciones: 100
- Probabilidad de cruce: 0’5 (entre 0 y 1)
- Probabilidad de mutación: 0’01 (entre 0 y 1)

En el caso del parámetro de tamaño de la población, se ha utilizado un tamaño de 100 individuos para los dominios: *Thyroides*, *Car* y *Balance-Scale*. Para el dominio BUPA se ha utilizado una población de 50 individuos.

4.1.- Dominio *BUPA*

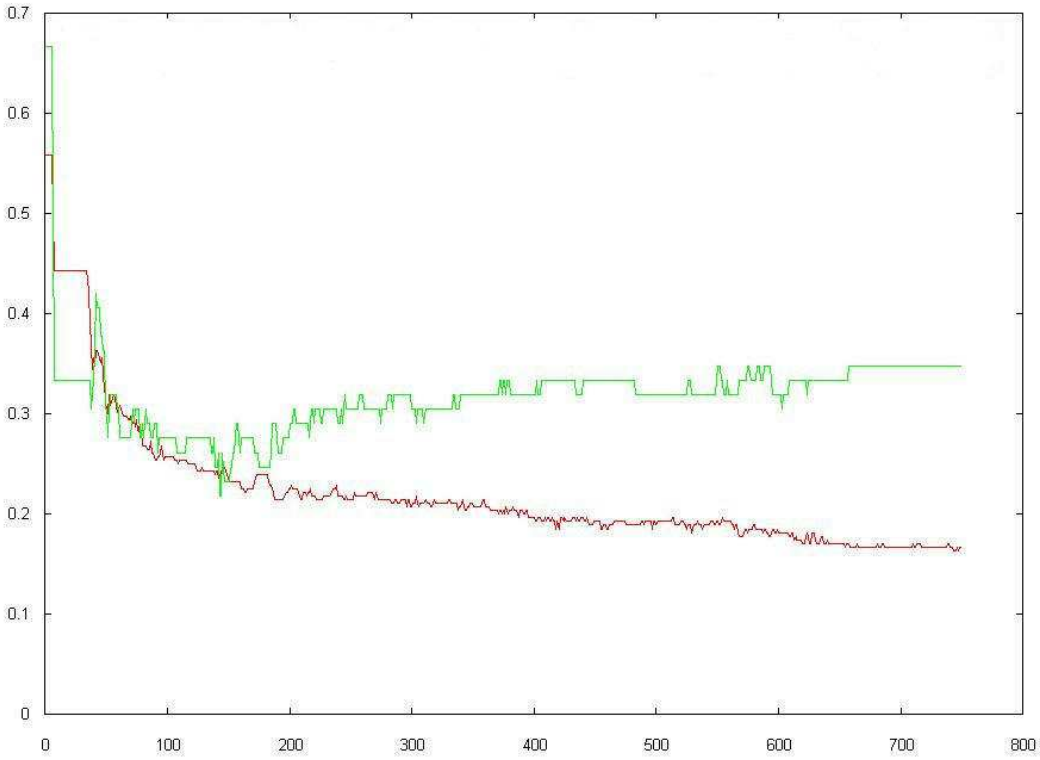
El verdadero nombre de este dominio es: ***Liver Disorders*** (Trastornos del Hígado), cuyos datos han sido suministrados por la BUPA (*British United Provident Association*) *Medical Research*. Consiste en 345 instancias, cada una de ellas recoge los datos pertenecientes a un varón.

Cada patrón (instancia) se compone de **6 atributos** y es catalogado entre **2 clases**. Los 5 primeros atributos son todas las pruebas de sangre que se cree que son sensibles a las alteraciones hepáticas que pudieran surgir por el consumo excesivo de alcohol. El sexto atributo se corresponde al consumo de alcohol diario, medido en unidades de media pinta (0’284 litros). La clasificación divide las instancias en aquellos individuos que padecen algún problema en el hígado y los que no. Las denominaremos ***clase1*** (que la conforman 145 patrones) y ***clase2*** (los 200 patrones restantes).

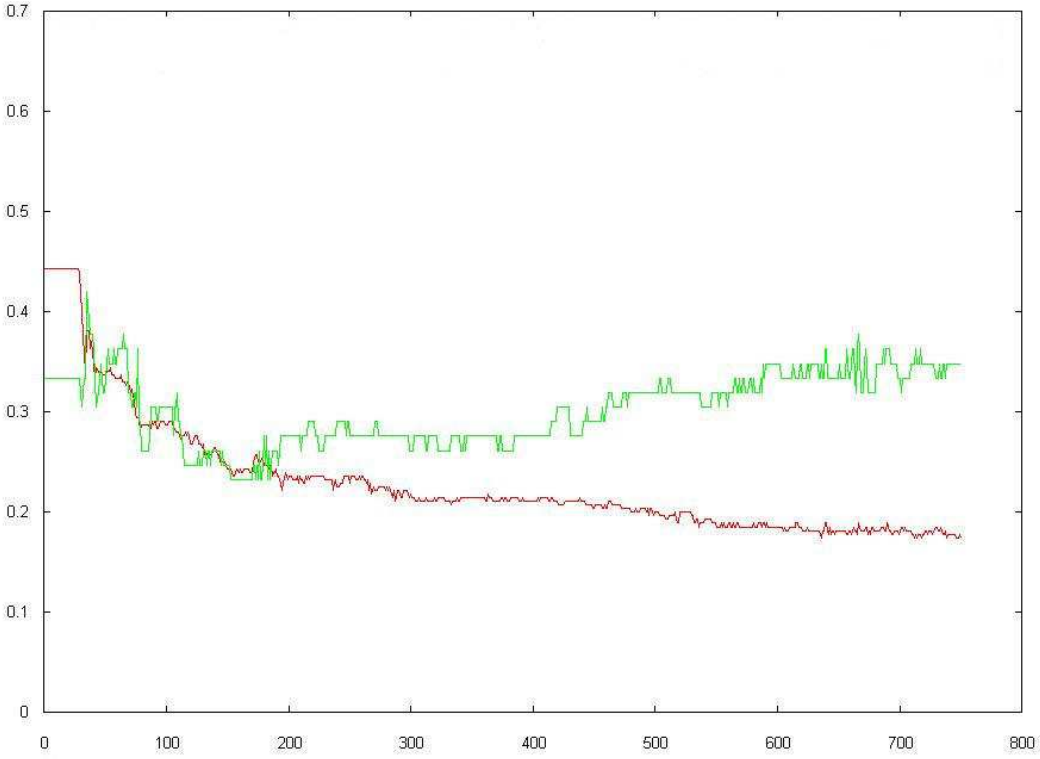
4.1.1.- Elección de la red

El conjunto de datos (375 patrones) se divide en dos partes, el **80%** es el conjunto de **entrenamiento** (276 patrones en este caso) y el **20%** restante el de **test** (69 patrones). Seguidamente se realizan dos ejecuciones durante 750 iteraciones y con una **razón de aprendizaje de 0’1**, una de ellas con una red de 15 neuronas ocultas y la otra con 25 neuronas ocultas.

A continuación, se muestran las gráficas correspondientes a las ejecuciones realizadas. Estas gráficas representan la proporción de error en la clasificación (eje Y) para el **entrenamiento** (**rojo**) y para el **test** (**verde**), a lo largo del número de iteraciones (eje X). Esto es, si por ejemplo la gráfica muestra un 0’15 de error, equivale a un 85% de aciertos en la clasificación. Todas las gráficas correspondientes a redes de neuronas que se muestren en los demás dominios, seguirán esta misma estructura.



Red con 15 neuronas ocultas



Red con 25 neuronas ocultas

Ambas redes tienen un comportamiento muy similar, pues el error de clasificación tanto en el conjunto de entrenamiento como en el de test, tiene un desarrollo semejante a lo largo de las iteraciones. Por tanto, se opta por la red con **15 neuronas ocultas**, pues el coste de computación es menor. La **razón de aprendizaje** será **0'1** y las redes se entrenarán durante **180 iteraciones**, pues a partir de las 200 iteraciones el error de test aumenta, por lo que se empieza a producir sobreaprendizaje.

4.1.2.- Método 1

Tras realizar la validación cruzada con el algoritmo evolutivo, se obtienen los resultados en cada *fold*. Estos son, los porcentajes de **aciertos globales y** para cada **clase:** para la **red inicial** (entrenada con el cjto. de entrenamiento pero sin replicar ningún patrón) y para la **red final** (entrenada replicando los patrones mal clasificados según el cromosoma del mejor individuo de la evolución, es decir aquel del frente de Pareto que posee mayor porcentaje de aciertos global sobre el cjto. de entrenamiento).

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento	**Red Inicial**	**Global**	77'536232	76'086957	77'173913	75'362319	76'086957
		Clase 1	61'061947	67'213115	61'320755	64'102564	65'573770
		Clase 2	88'957055	83'116883	87'058824	83'647799	84'415584
	Red Final	**Global**	80'434783	80'072464	80'072464	81'521739	79'710145
		Clase 1	71'681416	76'229508	64'150943	76'068376	75'409836
		Clase 2	86'503067	83'116883	90'000000	85'534591	83'116883
test	**Red Inicial**	**Global**	73'913043	76'811594	63'768116	76'811594	75'362319
		Clase 1	68'750000	56'521739	43'589744	60'914286	65'217391
		Clase 2	78'378378	86'956522	90'000000	87'804878	80'434783
	Red Final	**Global**	78'260870	75'362319	73'913043	63'768116	71'014493
		Clase 1	78'125000	56'521739	61'538462	60'714286	65'217391
		Clase 2	78'378378	84'782609	90'000000	65'853659	73'913043

Como se puede apreciar, el porcentaje de aciertos para el conjunto de entrenamiento mejora en todos los *folds*, sobre todo en la *clase1*, cuyo aumento es más significativo que en la *clase2*, la cual llega a disminuir en el *fold1* y en el *fold5*. Respecto al conjunto de test, sólo en algunos *folds* incrementa el porcentaje de aciertos (*fold1* y *fold3*). Comparando los porcentajes de aciertos por clase en el test, en la *clase1* se mejora el porcentaje en rasgos generales, mientras que en la *clase2* se devalúa. Para obtener una valoración general, se realiza la **media** de los **porcentajes globales** y por cada **clase** a partir de los porcentajes de cada *fold*.

Media		**entrenamiento**	**test**
Red Inicial	**Global**	**76'449276**	**73'333333**
	Clase 1	63'854430	58'998632
	Clase 2	85'439229	84'714912
Red Final	**Global**	**80'362319**	**72'461768**
	Clase 1	72'708016	64'423376
	Clase 2	85'654285	78'585538

En resumen, el porcentaje de aciertos en el cjto. de **entrenamiento mejora** con la replicación de los patrones mal clasificados (aprox. un 4%), pero en el cjto. de **test disminuye** casi un 1%. Esto puede deberse a un sobreaprendizaje, pero valorando también las anteriores observaciones y a los porcentajes por clase, se puede decir que la *clase2* no se adapta bien al modelo desarrollado, pues sufre grandes pérdidas en las tasas de acierto del test (disminuye en unos 6 puntos).

A continuación se muestra una gráfica de los frentes de Pareto que se obtienen durante la evolución del algoritmo, para las siguientes iteraciones:

Población inicial: ◊ (iteración 0)

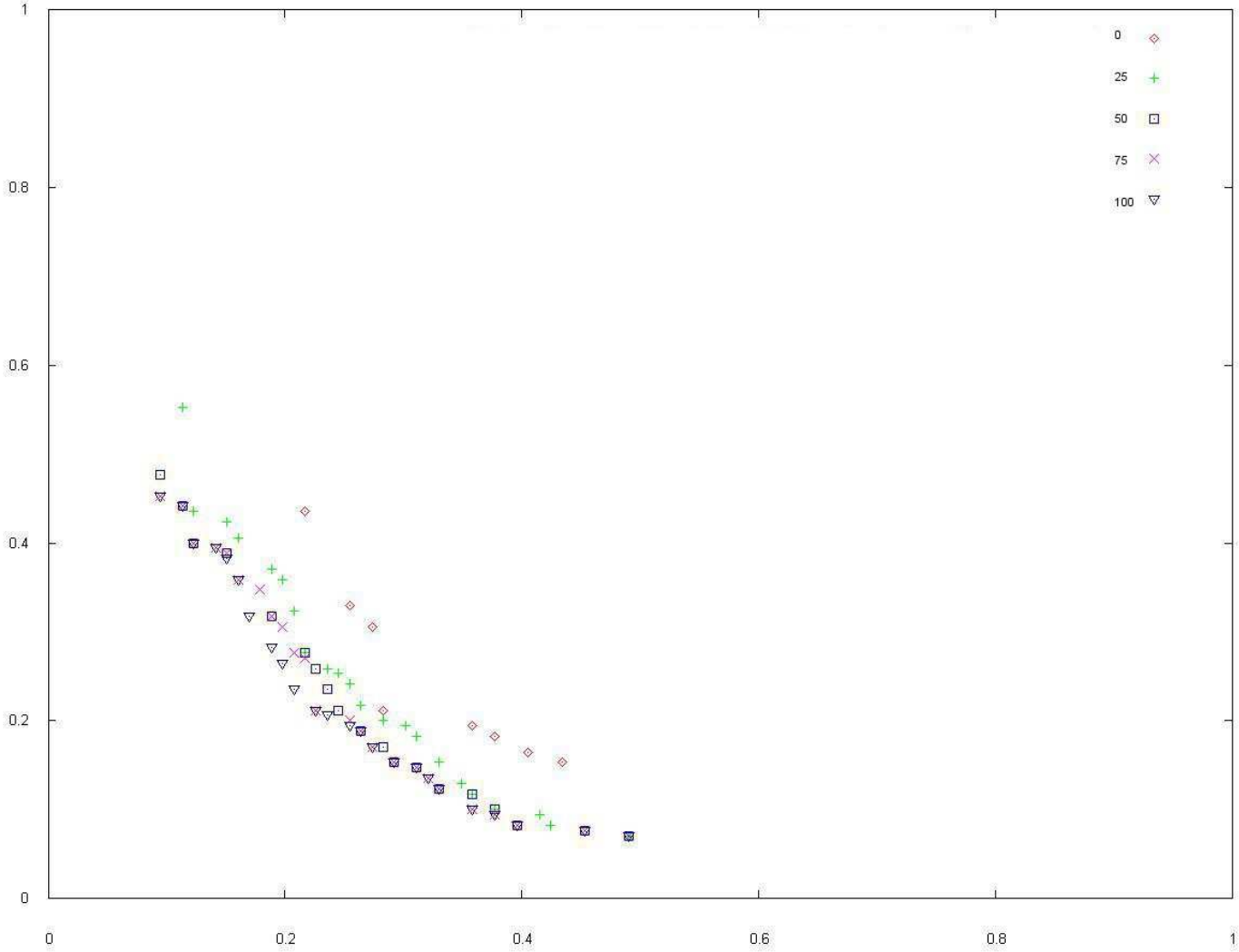
Iteración 25: +

Iteración 50: ▫

Iteración 75: ×

Iteración 100: ∇ (el frente de Pareto final del algoritmo)

Los valores mostrados son los porcentajes de error de cada clase (entre 0 y 1) que se producen en cada individuo del frente, a partir del conjunto de entrenamiento. La gráfica que se expone, corresponde con los frentes de Pareto del *fold3*, al ser la más descriptiva. Se observa que el frente evoluciona a lo largo de las iteraciones.



Frentes de Pareto para el fold3 en el método1

4.1.3.- Método 2

Como se explicó en el ***Apartado 3.3.1*** este método es semejante al anterior pero ahora el conjunto de entrenamiento original (276 patrones) está dividido en dos partes: el **80%** es ***entrenamiento1*** (221 patrones) y el **20%** restante es ***entrenamiento2*** (55 patrones). La primera de ellas se utilizará para entrenar la red (replicando los patrones pertinentes) y la segunda para evaluar la red tanto en el *fitness* como en la elección del mejor individuo del frente de Pareto. Por tanto, ahora también obtendremos un tercer grupo de porcentajes más.

En la siguiente tabla se muestran los porcentajes de aciertos para cada uno de los *folds*. Del mismo modo que en el anterior método, se incluye una tabla con las medias de los porcentajes de aciertos globales, para facilitar la interpretación de los resultados.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento1	**Red Inicial**	**Global**	78’280543	76’018100	75’113122	76’470588	74’660633
		Clase 1	64’835165	63’636364	60’674157	60’227273	67’307692
		Clase 2	87’692308	86’065574	84’848485	87’218045	81’196581
	Red Final	**Global**	68’778281	69’683258	76’018100	75’565611	75’565611
		Clase 1	49’450549	72’727273	62’921348	70’454545	69’230769
		Clase 2	82’307692	67’213115	84’848485	78’947368	81’196581
entrenamiento2	**Red Inicial**	**Global**	78’181818	72’727273	78’181818	74’545455	76’363636
		Clase 1	68’181818	65’217391	58’823529	62’068966	61’111111
		Clase 2	84’848485	78’125000	86’842105	88’461538	83’783784
	Red Final	**Global**	81’818182	85’454545	87’272727	90’909091	85’454545
		Clase1	59’090909	78’260870	64’705882	93’103448	61’111111
		Clase 2	96’969697	90’625000	97’368421	88’461538	97’297297
test	**Red Inicial**	**Global**	75’362319	75’362319	62’318841	73’913043	78’260870
		Clase 1	68’750000	56’521739	38’461538	53’571429	60’869565
		Clase 2	81’081081	84’782609	93’333333	87’804878	86’956522
	Red Final	**Global**	68’115942	65’217391	69’565217	60’869565	71’014493
		Clase 1	75’000000	69’565217	56’410256	71’428571	69’565217
		Clase 2	62’162162	63’043478	86’666667	53’658537	71’739130

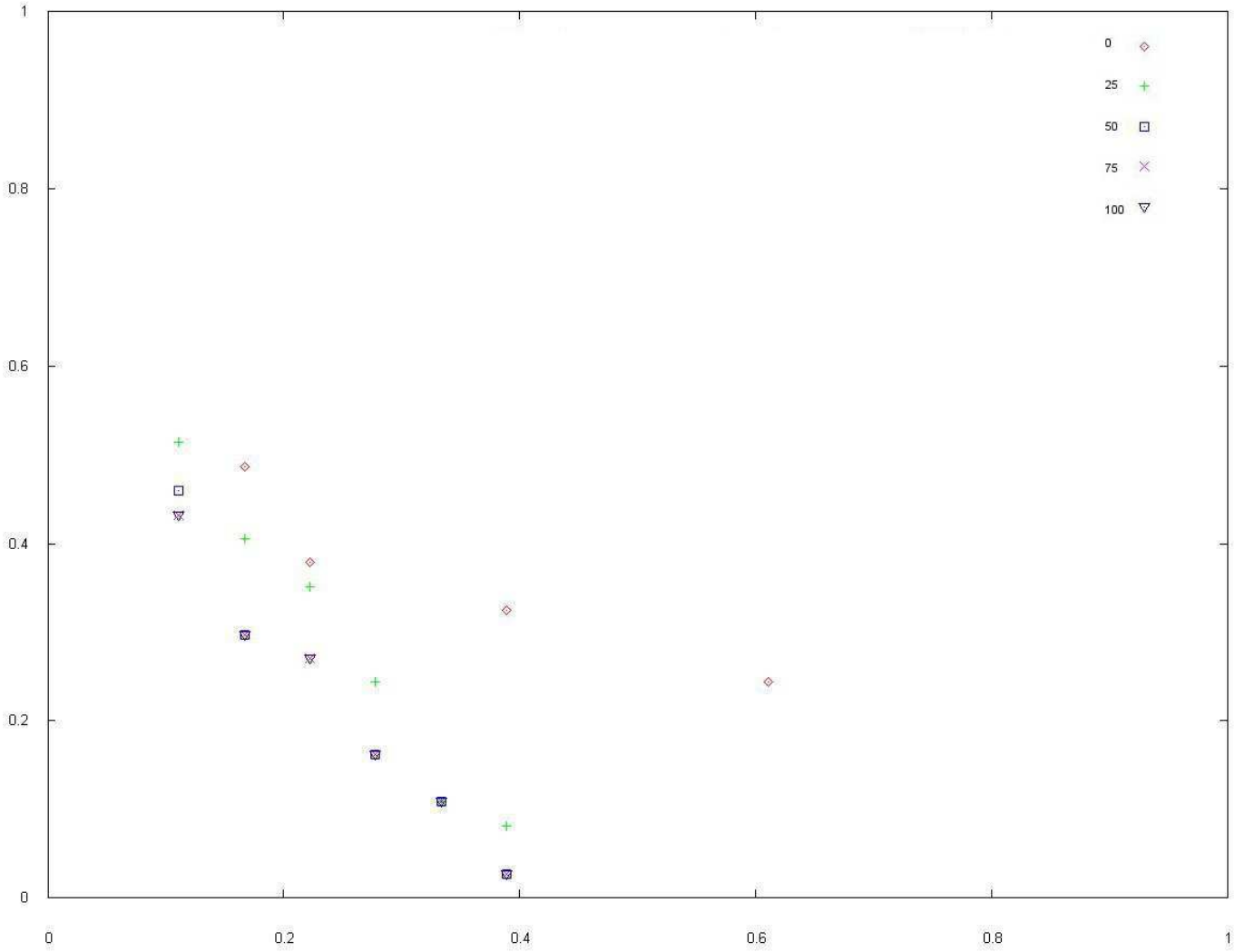
Media		**entrenamiento1**	**entrenamiento2**	**test**
Red Inicial	**Global**	**76’108597**	**76’000000**	**73’043478**
	Clase 1	63’336130	63’080563	55’634854
	Clase 2	85’404199	84’412182	86’791685
Red Final	**Global**	**73’122172**	**86’181818**	**66’956522**
	Clase 1	64’956897	71’254444	68’393852
	Clase 2	78’902648	94’144391	67’453995

Como se puede observar, el único porcentaje de aciertos que mejora es del conjunto de *entrenamiento2*, el resto empeoran. El de test empeora considerablemente, por lo que es un mal indicio. Además, esta disminución en el test es provocada únicamente por el gran decrecimiento en el porcentaje de aciertos en la *clase2*, pues en la *clase1* se mejora. Con el conjunto de *entrenamiento1* ocurre similar, pues en general hay una mejora en los aciertos en la *clase1* (a excepción del *fold1*), mientras que en la *clase2* se empeora o se mantiene el porcentaje de aciertos. Esto puede ser debido principalmente al escaso número de patrones de entrenamiento, lo cual dificulta el aprendizaje de la red e incluso podría producirse sobreaprendizaje.

El único *fold* en el que mejoran los aciertos con *entrenamiento1* y test, es el *fold3*. Sobre todo se mejora bastante en el test, como también ocurría en el anterior método, donde mejoraba considerablemente. Esto es debido al reparto que se realiza del conjunto total de patrones para este *fold*, que se habrán distribuido de una forma favorable a la aplicación desarrollada. También hay que resaltar, que en el resto de *folds*, cuanto mayor es la mejora con el conjunto de *entrenamiento2*, mayor es el empeoramiento del porcentaje de aciertos con el conjunto de test.

En definitiva, el algoritmo se sobreadapta al conjunto de *entrenamiento2*, que es el utilizado para evaluar las redes y por tanto es el que encauza el algoritmo genético a la hora de valorar el individuo según el número de replicas que debe de haber de cada patrón. Sobre todo, esta sobreadaptación es mayor en la *clase2*, pues en todos los *folds* se produce una gran mejoría (con el conjunto de *entrenamiento2*), mientras que con la *clase1* incluso llega a empeorar en el *fold1*.

La siguiente gráfica muestra los distintos frentes de Pareto que se producen durante la evolución del algoritmo en el *fold5* a lo largo de las generaciones. En este caso el conjunto de entrenamiento utilizado para calcular el frente es *entrenamiento2*.



Frentes de Pareto para el fold5 en el método2

4.1.4.- Método 3

Como se describió en el ***Apartado 3.3.2***, este método es similar al método 1, con la diferencia que los **patrones difusos** son los que se replican en vez de los mal clasificados. A continuación se muestran los porcentajes de acierto obtenidos para la red inicial y final en cada uno de los *folds* y para los conjuntos de entrenamiento y de test, diferenciando también cada una de las clases. Asimismo, se muestran las medias de los porcentajes obtenidos entre todos los *folds*.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento	**Red Inicial**	**Global**	77’536232	76’086957	77’173913	75’362319	76’086957
		Clase 1	61’061947	67’213115	61’320755	64’102564	65’573770
		Clase 2	88’957055	83’116883	87’058824	83’647799	84’415584
	Red Final	**Global**	80’072464	80’072464	81’159420	82’246377	80’797101
		Clase 1	65’486726	71’311475	65’094340	76’068376	73’770492
		Clase 2	90’184049	87’012987	91’176471	86’792453	86’363636
test	**Red Inicial**	**Global**	73’913043	76’811594	63’768116	76’811594	75’362319
		Clase 1	68’750000	56’521739	43’589744	60’714286	65’217391
		Clase 2	78’378378	86’956522	90’000000	87’804878	80’434783
	Red Final	**Global**	72’463768	72’463768	65’217391	65’217391	71’014493
		Clase 1	56’250000	56’521739	43’589744	60’714286	69’565217
		Clase 2	86’486486	80’434783	93’333333	68’292683	71’739130

Media		**entrenamiento**	**test**
Red Inicial	**Global**	**76’4492756**	**73’3333332**
	Clase 1	63’854430	58’958632
	Clase 2	85’439229	84’714912
Red Final	**Global**	**80’8695652**	**69’2753622**
	Clase 1	70’346282	57’328197
	Clase 2	88’305919	80’057283

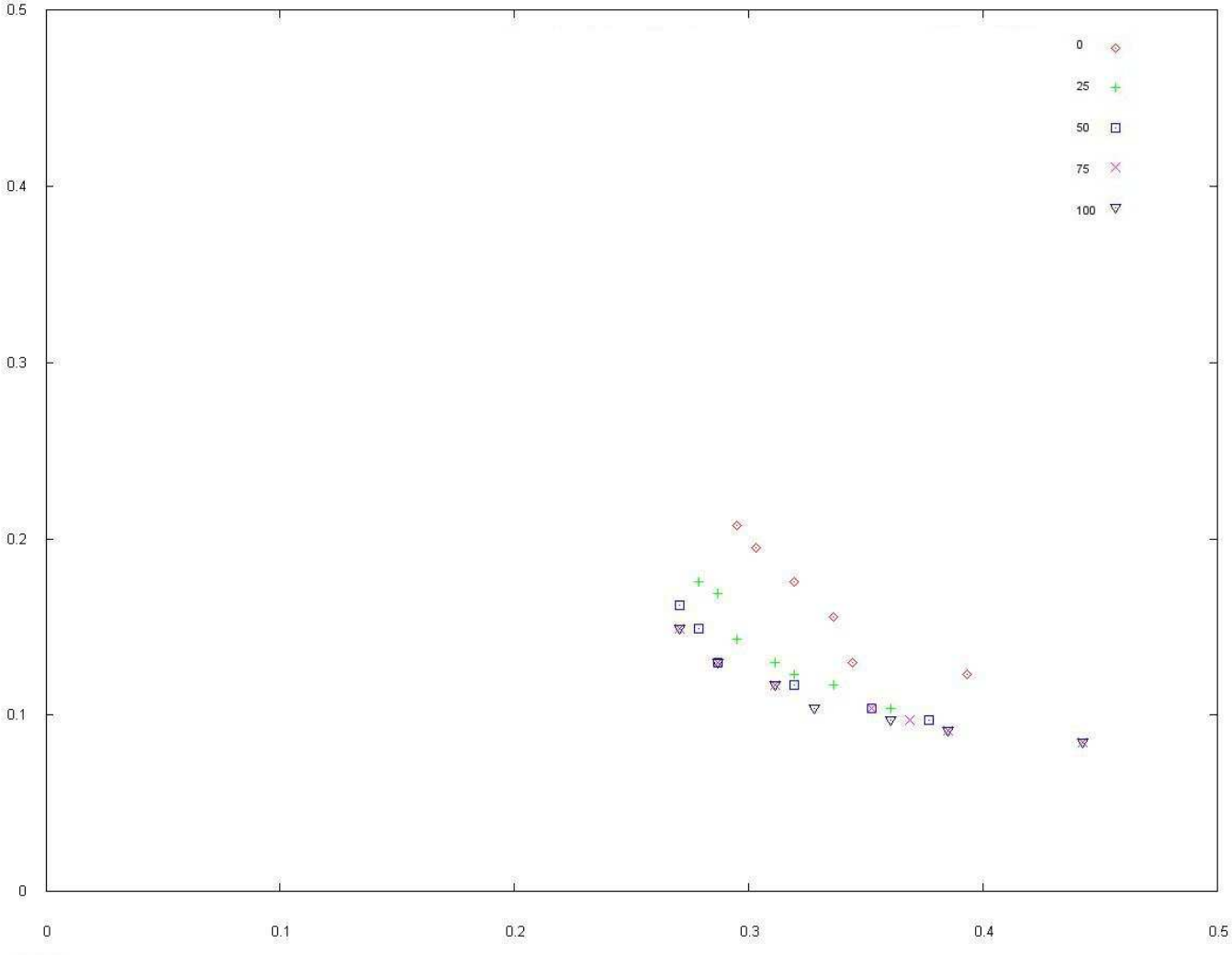
En este método, se dan unos resultados similares que con el método 1. El porcentaje de aciertos mejora con el conjunto de entrenamiento, pero en el conjunto de test disminuye bastante, incluso más que con el método 1. En el conjunto de entrenamiento, los aciertos aumentan en todos los *folds* para las dos clases. En el conjunto de test, la media de porcentaje de aciertos para la *clase1* se devalúa en 1’63 puntos, y en la *clase2* disminuye en 4’66 puntos. Por tanto, en el conjunto de test empeoran más los resultados en la *clase2* que en la *clase1*, como ocurría en los anteriores métodos.

Para cada uno de los *folds* se ha elaborado una tabla que relaciona los dos criterios (en la selección de los patrones a replicar) que se han utilizado entre el método 1 y el método 3. En esta tabla se muestra cuantos patrones del conjunto de entrenamiento han sido considerados difusos y los que no, junto con cuantos patrones han sido bien y mal clasificados. Todo ello es calculado al inicio del algoritmo con la red inicial. Esta tabla ayudará a comparar ambos métodos. Para simplificar, se incluirá una tabla con las medias de los distintos valores de la tabla de cada *fold*.

	Bien clasificados	Mal clasificados	
Difusos	13’6	13’6	**27’2**
No difusos	197’4	51’4	**248’8**
	211	**65**	**276**

En primer lugar se puede reseñar que hay menos patrones difusos que patrones mal clasificados. Por tanto, en este método el cromosoma será más pequeño que en el método 1. Hay que resaltar la gran cantidad de patrones mal clasificados que no han sido considerados como difusos. Esto seguramente sea la razón de que se logren peores resultados que con el método 1. También es reseñable que el conjunto de patrones difusos está compuesto en igual proporción de patrones bien y mal clasificados.

A continuación se muestra una gráfica que presenta los distintos frentes de Pareto que se producen durante la evolución del algoritmo en el *fold2*.



Frentes de Pareto para el fold2 en el método3

4.1.5.- Análisis

En primer lugar, hay que resaltar que este dominio tiene muy pocos datos, y esto provoca que haya un sobreaprendizaje del algoritmo, lo que conlleva que empeoren los resultados obtenidos con el conjunto de test.

A continuación se muestra una tabla con los tres métodos y las medias de los porcentajes para la red inicial y final, para facilitar la comparación entre los distintos métodos. Nombraremos al conjunto de entrenamiento como *train*, y se ha redondeado a cuatro decimales.

Red	**Método 1**		**Método 2**			**Método 3**	
	train	***test***	***train1***	***train2***	***test***	***train***	***test***
Inicial	76’4493	73’3333	76’1086	76’0000	73’0435	76’4493	73’3333
Final	80’3623	72’4618	73’1222	86’1818	66’9565	80’8696	69’2754

Para el conjunto de test (que es el mismo en los tres métodos), aquel método donde más empeora es el 2, seguido por el 3. Por lo que el *método1*, aunque también empeore, es el mejor para este dominio. Respecto al conjunto de entrenamiento, es difícil hacer una comparación con el *método2*, pues éste lo divide en dos partes. Si lo comparamos con *entrenamiento1*, en el *método2* da el peor resultado, y comparándolo con *entrenamiento2*, proporciona el mejor resultado. Comparando el entrenamiento entre el *método1* y el *método3*, éste último mejora un poco más el porcentaje de aciertos.

Para comprender mejor el por qué de que se produzca la mayor mejora en el *fold3* con el conjunto de test, mientras que en el resto de *folds* en general empeora, se ha observado el reparto de patrones para cada *fold* en los conjuntos de entrenamiento (*train*) y test para cada una de las clases.

		fold1	***fold2***	***fold3***	***fold4***	***fold5***
train	***clase1***	113	122	106	117	122
	clase2	163	154	170	159	154
	clase2 – clase1	+ 50	+ 32	+ 64	+ 42	+ 32
test	***clase1***	32	23	39	28	23
	clase2	37	46	30	41	46
	clase2 – clase 1	+ 5	+ 23	– 9	+ 13	+ 23

Como se puede apreciar, en el *fold3* hay un mayor número de patrones de la *clase2* en el conjunto de entrenamiento, en comparación con el resto de *folds*, y en el conjunto de test ocurre lo contrario, hay muchos más patrones de la *clase1* y menos de la *clase2*. Con esto y los estudios que se han ido realizando anteriormente en los distintos métodos, se puede concluir que la *clase2* es la que peor se adapta a nuestro modelo desarrollado. Por tanto, si en el entrenamiento hay muchos más patrones (como es el caso de *fold3)*, la aplicación tendrá un mejor aprendizaje. Del mismo modo, al haber más patrones de la *clase2* en entrenamiento, el test tendrá menos de ésta, por lo que también se verá reflejado en que habrá más aciertos en la evaluación.

4.2.- Dominio *Car*

Este dominio se corresponde con los datos ***Car Evaluation*** (evaluación de coches) del repositorio UCI. Este dominio contiene 1.728 instancias que recoge las características de un vehículo (precio, capacidad, etc.), y el grado de aceptación del coche.

Cada instancia (patrón) está compuesto por **6 atributos** (las características del coche) y puede ser catalogado entre **4 clases** (el nivel de aceptación del vehículo). Los atributos se corresponden con las siguientes características del vehículo:

- Precio de compra: *muy-alto, alto, medio, bajo*
- Precio del mantenimiento: *muy-alto, alto, medio, bajo*
- Número de puertas: *2, 3, 4, 5-ó-más*
- Número de pasajeros: *2, 4, más*
- Tamaño del maletero: *pequeño, medio, grande*
- Seguridad estimada: *baja, media, alta*

El sistema utiliza valores numéricos como entrada (los atributos), que serán las entradas a las redes de neuronas que se generarán. Por tanto, es necesario transformar los valores nominales de los atributos a valores numéricos:

- Precio (de compra o de mantenimiento):
  - *muy-alto*: 0’9
  - *alto*: 0’63
  - *medio*: 0’36
  - *bajo*: 0’1
- Puertas:
  - *2*: 0’1
  - *3*: 0’36
  - *4*: 0’63
  - *5-o-más*: 0’9
- Pasajeros:
  - *2*: 0’1
  - *4*: 0’5
  - *más*: 0’9
- Maletero/Seguridad:
  - *pequeño/baja*: 0’1
  - *medio/a*: 0’5
  - *grande/alta*: 0’9

La clasificación de las instancias es según el nivel de aceptación que tiene el vehículo en cuestión. Este nivel de aceptación se divide en 4 grados, que se corresponden con las 4 clases en las que se puede clasificar un patrón. Estas clases son también nominales, y en nuestro sistema la salida es numérica, por tanto:

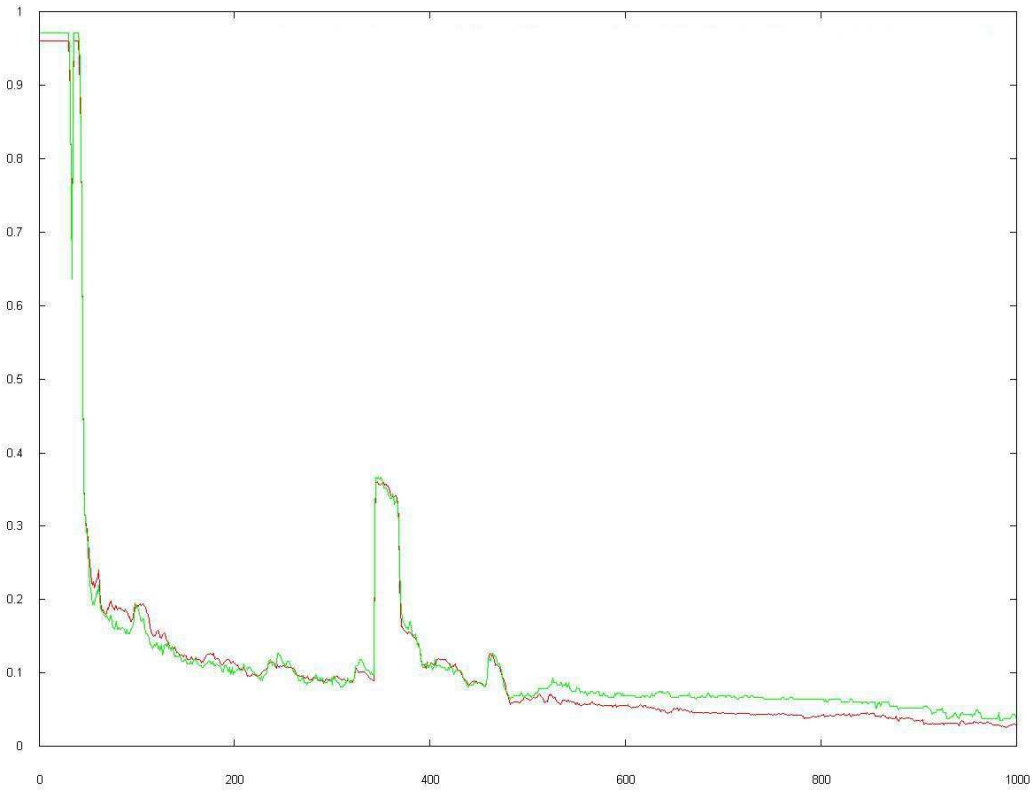
- inaceptable: *clase1* (1.210 patrones)
- aceptable: *clase2* (384 patrones)
- bueno: *clase3* (69 patrones)
- muy-bueno: *clase4* (65 patrones)

4.2.1.- Elección de la red

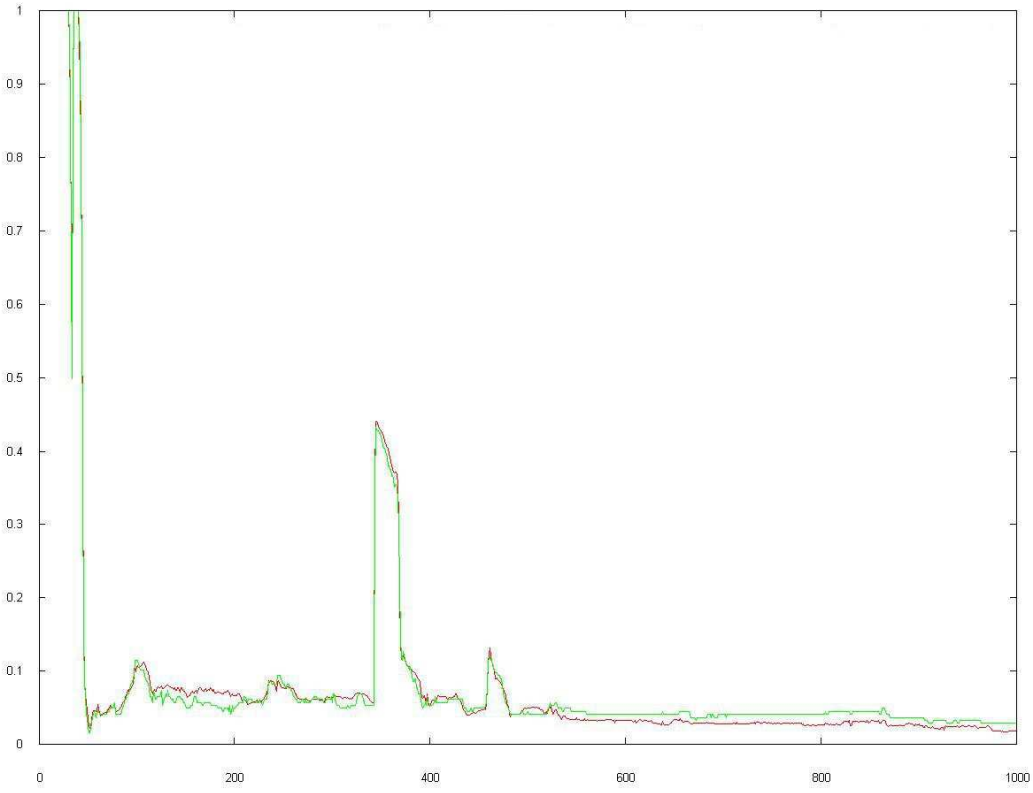
Al igual que en el anterior dominio, el conjunto de datos (1.728 patrones) se ha dividido en dos partes, el **80%** para **entrenamiento** (1.382 patrones en este caso) y el **20%** para **test** (346 patrones). Se han realizado varias pruebas para la selección de la mejor arquitectura de la red de neuronas para este dominio. Se ha probado con:

- Razón de aprendizaje: 0'1 y 0'05
- Nº de neuronas ocultas: 15, 20 y 25

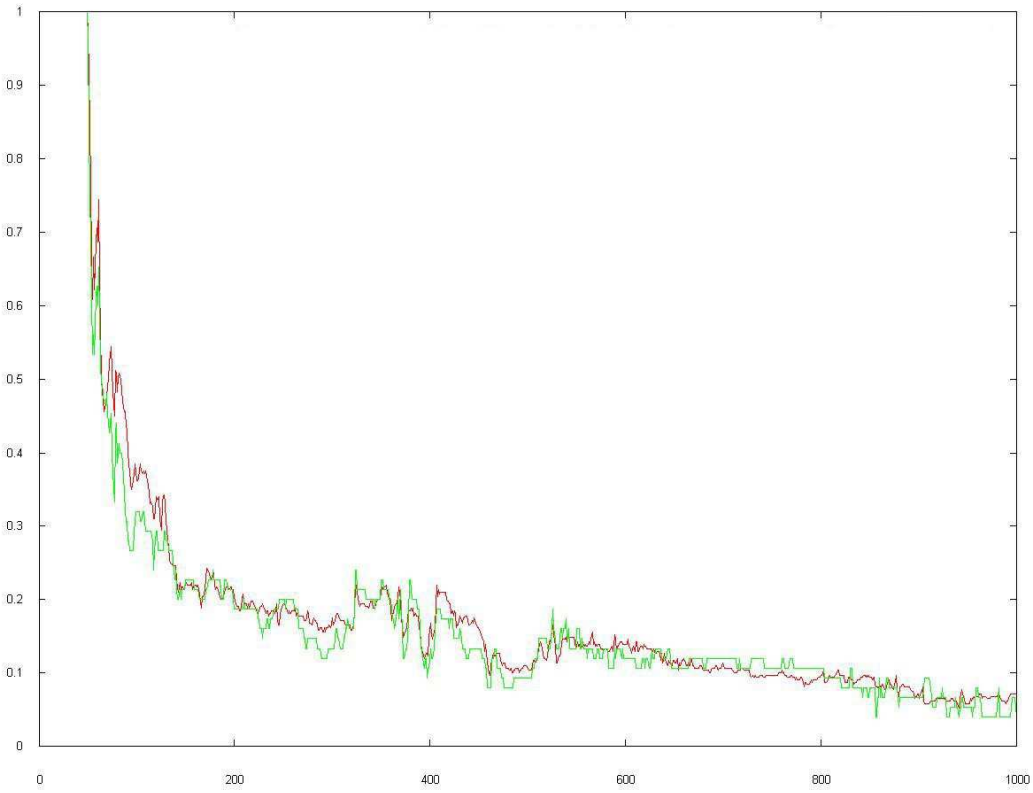
Finalmente, se ha seleccionado **0'05** de **razón de aprendizaje** y **20 neuronas ocultas**. Se ha entrenado la red durante **1000 iteraciones**, obteniendo los errores de clasificación (global y por cada clase) para el entrenamiento (rojo) y para el test (verde), generando unas gráficas del mismo modo que en el dominio BUPA.



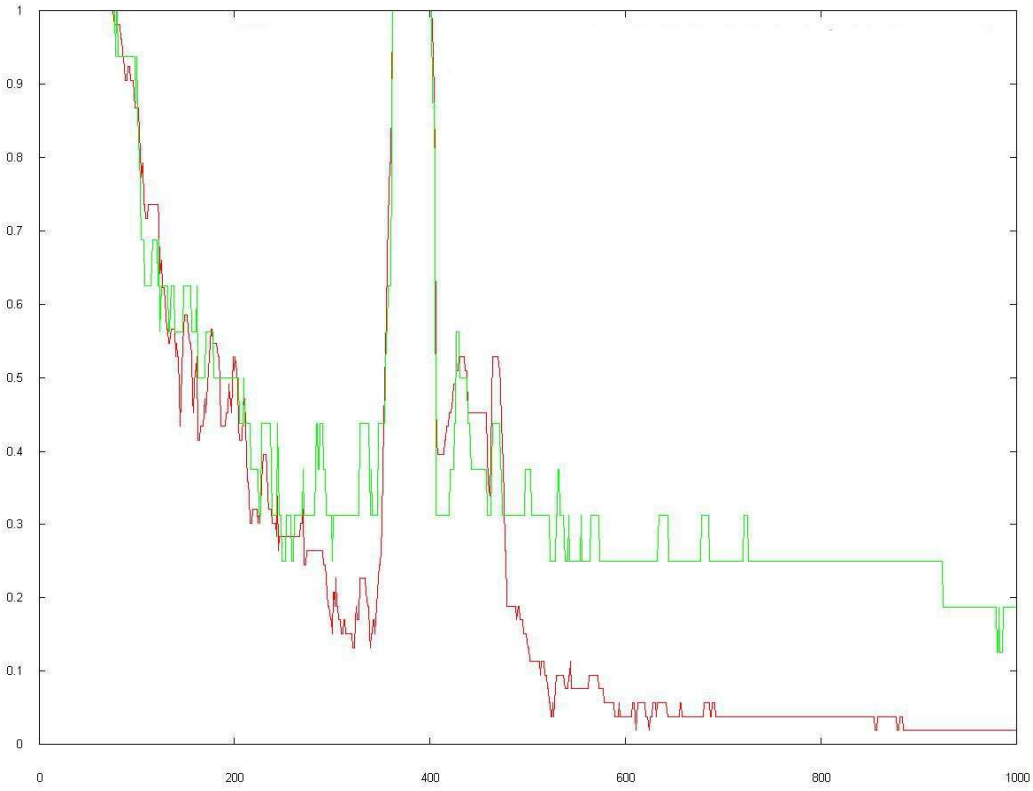
Error de clasificación global



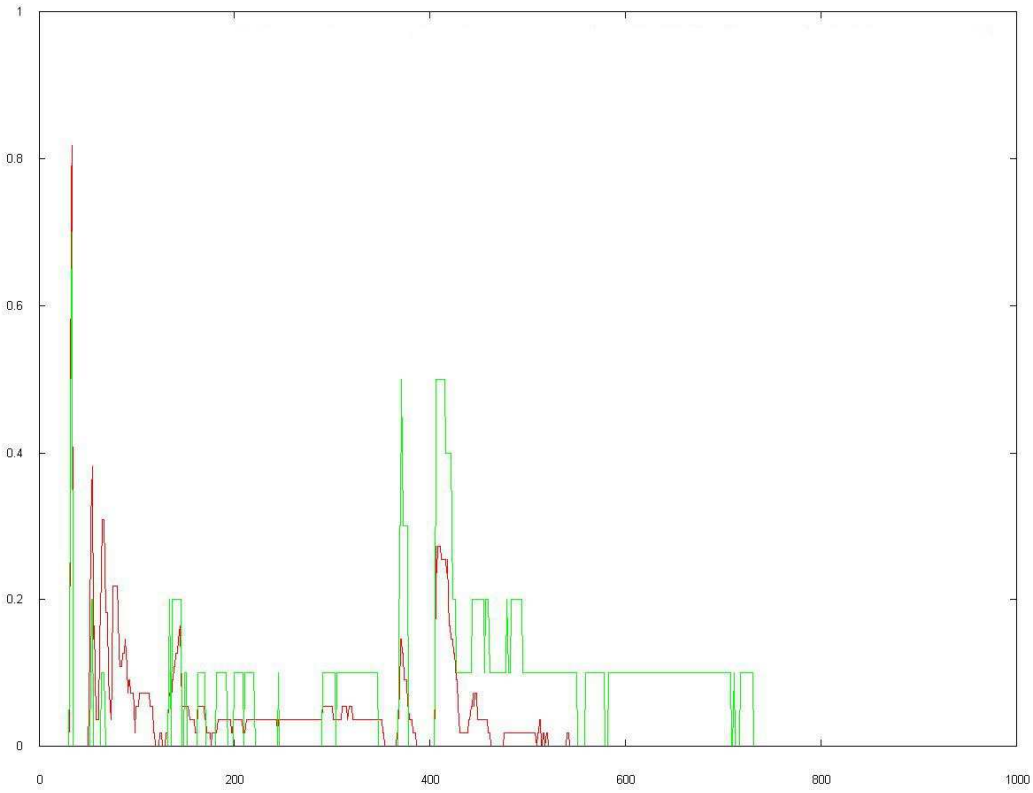
Error de clasificación de la clase1



Error de clasificación de la clase2



Error de clasificación de la clase3



Error de clasificación de la clase4

Observando el error de clasificación global se puede apreciar que la red de neuronas obtiene muy buenos resultados. Sin embargo, al observar el error de clasificación de cada clase se distingue que la red no se adapta tan bien para aquellas clases que tienen pocos patrones (como las clases 2 y 3).

Al ser un dominio con datos *desequilibrados*, la red aprende mejor la clase que tiene un mayor número de patrones (*clase1*), mientras que en las que hay poco patrones, los llega a desestimar. Por tanto, es interesante comprobar si con el algoritmo implementado se puede lograr que la red de neuronas abarque más patrones de las clases más pequeñas.

4.2.2.- Método 1

A continuación se muestran los resultados obtenidos tras aplicar la validación cruzada con el algoritmo evolutivo, de igual forma que se explicó en el ***Apartado 3.2***. Se mostrarán los porcentajes de aciertos globales y para cada clase, para la red inicial y para la red final (el mejor punto del frente de Pareto). También se ha realizado la media de los porcentajes de aciertos que se obtienen entre todos los *folds*. El conjunto de entrenamiento lo conforman 1.382 patrones y el de test 346 patrones.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento	**Red Inicial**	**Global**	97’686189	95’224313	96’529284	98’408104	96’960926
		Clase 1	98’546210	97’435897	97’414685	99’072165	98’031088
		Clase 2	94’533762	86’798680	92’880259	95’723684	92’880259
		Clase 3	98’245614	98’181818	98’148148	100’00000	98’113208
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	99’276934	99’493488	99’421547	99’204052	99’493488
		Clase 1	99’688474	99’794872	99’896587	99’484536	99’689119
		Clase 2	97’749196	98’349835	97’734628	98’026316	98’705502
		Clase 3	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
test	**Red Inicial**	**Global**	97’391304	92’774566	95’942029	97’398844	96’242775
		Clase 1	97’570850	97’872340	97’530864	98’333333	97’142857
		Clase 2	95’890411	80’246914	92’000000	93’750000	96’000000
		Clase 3	100’00000	78’571429	86’666667	100’00000	81’250000
		Clase 4	100’00000	93’750000	100’00000	100’00000	100’00000
	Red Final	**Global**	97’391304	97’976879	98’550725	97’687861	97’976879
		Clase 1	97’165992	100’00000	100’00000	97’500000	99’183673
		Clase 2	97’260274	91’358025	94’666667	97’500000	94’666667
		Clase 3	100’00000	100’00000	93’333333	100’00000	93’750000
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000

Media		**entrenamiento**	**test**
Red Inicial	**Global**	**96’96176**	**95’94990**
	Clase 1	98’10001	97’69005
	Clase 2	92’56333	91’57747
	Clase 3	98’53776	89’29762
	Clase 4	100’0000	98’75000
Red Final	**Global**	**99’37790**	**97’91673**
	Clase 1	99’71072	98’76993
	Clase 2	98’11310	95’09033
	Clase 3	100’0000	97’41667
	Clase 4	100’0000	100’0000

En rasgos generales, se puede observar que se aumenta la precisión tanto en el conjunto de entrenamiento (en 2’42 puntos) como en el conjunto de test (en 1’97 puntos). En este dominio aumenta significativamente el porcentaje de aciertos para el conjunto de test, que es independiente del proceso evolutivo y de entrenamiento de la red, por lo que se obtienen unos resultados satisfactorios.

En todos los *folds* se produce un aumento del porcentaje de aciertos para todas las clases en el conjunto de entrenamiento. Con el conjunto de test, en general también aumenta para todas las clases, exceptuando el *fold1* y el *fold4*, en los que se produce una ligera disminución en la tasa de aciertos para la *clase1*. Esto es comprensible, pues para lograr cubrir mejor las clases minoritarias, es razonable que la tasa de aciertos de la clase mayoritaria disminuya.

También se puede contemplar que en la *clase4* siempre se llega a conseguir un 100% de aciertos. Los mayores aumentos que se producen son en la *clase3* (en el test hay un incremento de casi 8 puntos), seguida de la *clase2* (incrementa en 5'5 en entrenamiento y 2'5 en test). Por tanto, el grado de aumento del abarque de las clases minoritarias es superior al grado en el que puede llegar a disminuir la clase minoritaria, que aunque esto se produzca en algunos *folds*, en la media realizada aumenta.

En este dominio se dan 4 clases, por lo que la evolución de los frentes no puede ser mostrada a modo de gráfica, como en el anterior método. Por tanto, se ha realizado una tabla con los porcentajes de aciertos para cada clase del frente de Pareto final del algoritmo evolutivo para cada uno de los *folds*. Estos porcentajes se obtienen al evaluar el conjunto de entrenamiento con cada una de las redes finales (individuos) que conforman el frente de Pareto.

Frente de Pareto		***clase1***	***clase2***	***clase3***	***clase4***
fold1	*indiv1*	99'8962	95'1768	100	100
	indiv2	99'6885	97'7492	100	100
	indiv3	99'4808	98'0707	100	100
fold2	*indiv1*	99'6923	98'6799	98'1818	100
	indiv2	100	0	0	0
	indiv3	99'8974	97'0297	100	100
	indiv4	99'7949	98'3498	100	100
fold3	*indiv1*	99'8966	97'7346	100	100
fold4	*indiv1*	99'4845	98'0263	100	100
	indiv2	99'6907	97'6974	98'2456	100
	indiv3	99'7938	96'0526	100	100
fold5	*indiv1*	99'8964	94'822	100	100
	indiv2	99'6891	98'7055	100	100
	indiv3	99'7927	96'1165	100	100

De los 100 individuos que consta la población del algoritmo evolutivo, los frentes de Pareto de los *folds* resultan al final compuestos por muy pocos individuos. Lo que sucede es que de los 100 individuos que aparecen al final del algoritmo evolutivo, son repeticiones de estos conjuntos de valores, incluido también el cromosoma que no se muestra en la tabla por su gran tamaño. Incluso en el *fold3* sólo hay un único individuo. Por tanto, hay que resaltar el reducido tamaño del frente de Pareto resultante del algoritmo evolutivo.

4.2.3.- Método 2

En este método el conjunto de entrenamiento original (1.382 patrones) es dividido en dos partes: el **80%** es ***entrenamiento1*** (1.106 patrones) y el **20%** restante es ***entrenamiento2*** (276 patrones). La primera de ellas se utilizará para entrenar la red (replicando los patrones pertinentes) y la segunda para evaluar la red tanto en el *fitness* como en la elección del mejor individuo del frente de Pareto. Los resultados obtenidos se muestran en las dos siguientes tablas, en la primera separando para cada *fold* los porcentajes de aciertos globales y en cada clase, y en la segunda las medias de dichos porcentajes realizadas sobre todos los *folds*.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento1	**Red Inicial**	**Global**	97’651310	98’101266	96’567299	97’920434	95’660036
		Clase 1	99’215686	99’743590	97’458704	99’485861	97’689345
		Clase 2	92’549020	92’244898	92’372881	92’276423	87’966805
		Clase 3	97’674419	100’00000	100’00000	100’00000	97’560976
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	99’367660	98’824593	97’922313	98’372514	99’095841
		Clase 1	99’869281	99’487179	99’237611	99’100257	99’358151
		Clase 2	97’647059	96’326531	92’796610	95’528455	97’925311
		Clase 3	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
entrenamiento2	**Red Inicial**	**Global**	96’014493	97’463768	93’840580	96’014493	93’478261
		Clase 1	98’484848	98’461538	96’666667	98’437500	95’698925
		Clase 2	87’500000	93’103448	84’931507	86’206897	86’764706
		Clase 3	92’857143	100’00000	100’00000	100’00000	91’666667
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	99’275362	98’913043	97’826087	99’275362	100’00000
		Clase1	99’494949	98’461538	100’00000	99’479167	100’00000
		Clase 2	100’00000	100’00000	93’150685	98’275862	100’00000
		Clase 3	92’857143	100’00000	91’666667	100’00000	100’00000
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
test	**Red Inicial**	**Global**	97’101449	97’109827	94’782609	96’820809	94’219653
		Clase 1	98’785425	100’00000	97’119342	97’500000	95’510204
		Clase 2	91’780822	87’654321	85’333333	93’750000	89’333333
		Clase 3	91’666667	100’00000	100’00000	100’00000	93’750000
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	98’840580	97’398844	97’101449	97’687861	97’687861
		Clase 1	98’785425	98’461538	98’765432	99’166667	99’183673
		Clase 2	100’00000	100’00000	92’000000	92’500000	94’666667
		Clase 3	100’00000	100’00000	93’333333	100’00000	93’750000
		Clase 4	92’307692	100’00000	100’00000	100’00000	90’000000

Media		entrenamiento1	entrenamiento2	test
Red Inicial	**Global**	**97’180069**	**95’362319**	**96’006869**
	Clase 1	98’718637	97’549896	97’782994
	Clase 2	91’482005	87’701312	89’570362
	Clase 3	99’047079	96’904762	97’083333
	Clase 4	100’00000	100’00000	100’00000
Red Final	**Global**	**98’716584**	**99’057971**	**97’743319**
	Clase 1	99’410496	99’487131	98’872547
	Clase 2	96’044793	98’285309	95’833333
	Clase 3	100’00000	96’904762	97’416667
	Clase 4	100’00000	100’00000	96’461538

Para todos los conjuntos utilizados se produce un aumento en el porcentaje de aciertos global en todos los *folds*, y en base a las medias realizadas se observa un aumento de 1’537 puntos para *entrenamiento1*, de 3’696 puntos para *entrenamiento2* y 1’736 en el conjunto de test.

En comparación con el anterior dominio, ahora se consigue una mejora en todos los conjuntos, mientras que en *BUPA* sólo se producía con *entrenamiento2*. Pero es reseñable que en este dominio la mayor subida se origina en *entrenamiento2*. Este conjunto es el utilizado para evaluar las redes en este método, y por tanto es el que encamina el algoritmo evolutivo.

En definitiva, los resultados obtenidos son muy buenos, aunque no tanto como los obtenidos con el método anterior. Sobre todo, esto se aprecia al comparar el aumento de la tasa de aciertos del método anterior con el actual. Una posible causa de que no se consigan unos mejores resultados que con el *metodo1*, puede ser la disminución del conjunto de entrenamiento de la red (en este método denominado *entrenamiento1*).

Al igual que en el anterior método, se han recogido los porcentajes de aciertos para cada clase (con el conjunto de *entrenamiento2*) de los individuos que conforman el frente de Pareto resultante en cada *fold*.

Frente de Pareto		***clase1***	***clase2***	***clase3***	***clase4***
fold1	*indiv1*	99’4949	100	92’8571	100
	indiv2	100	92’8571	78’5714	100
	indiv3	100	96’4286	85’7143	87’5
	indiv4	98’9899	100	100	100
fold2	*indiv1*	98’4615	100	100	100
fold3	*indiv1*	100	90’411	100	100
	indiv2	98’8889	94’5205	91’6667	100
	indiv3	100	93’1507	91’6667	100
fold4	*indiv1*	99’4792	98’2759	100	100
	indiv2	100	91’3793	93’3333	100
fold5	*indiv1*	100	100	100	100

Del mismo modo que en el *metodo1*, el frente de Pareto que resulta del algoritmo genético es muy reducido, llegando incluso a contener solamente a un individuo, como es el caso de *fold2* y de *fold5*.

4.2.4.- Método 3

En este método se replican los patrones difusos en vez de los mal clasificados, tal y como se ha especificado en el ***Apartado 3.3.2***. Este método fue creado para comprobar si se podían mejorar el resultado en el dominio anterior BUPA. Aún a pesar de que no se mejoraron los resultados en ese dominio, se ha decidido seguir utilizándolo con el resto de dominios.

A continuación se muestran los porcentajes de acierto obtenidos para la red inicial y final en cada uno de los *folds* y para los conjuntos de entrenamiento y de test, diferenciando también cada una de las clases. Asimismo, se muestran las medias de los porcentajes obtenidos entre todos los *folds*. También se indica una tabla que relaciona los dos criterios utilizados para la replicación de los patrones: *difusión* y *clasificación*; que resulta de las medias de dichos valores de cada *fold*.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento	**Red Inicial**	**Global**	97’686189	95’224313	96’529284	98’408104	96’960926
		Clase 1	98’546210	97’435897	97’414685	99’072165	98’031088
		Clase 2	94’533762	86’798680	92’880259	95’723684	92’880259
		Clase 3	98’245614	98’181818	98’148148	100’00000	98’113208
		Clase 4	100’00000	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	99’204628	99’131693	98’915401	98’408104	99’131693
		Clase 1	99’480789	99’487179	99’586350	99’072165	99’585492
		Clase 2	98’070740	98’019802	96’440129	95’723684	97’411003
		Clase 3	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 4	100’00000	97’959184	100’00000	100’00000	100’00000
test	**Red Inicial**	**Global**	97’391304	92’774566	95’942029	97’398844	96’242775
		Clase 1	97’570850	97’872340	97’530864	98’333333	97’142857
		Clase 2	95’890411	80’246914	92’000000	93’750000	96’000000
		Clase 3	100’00000	78’571429	86’666667	100’00000	81’250000
		Clase 4	100’00000	93’750000	100’00000	100’00000	100’00000
	Red Final	**Global**	98’260870	95’375723	98’260870	97’398844	98’265896
		Clase 1	97’975709	100’00000	99’588477	98’333333	98’775510
		Clase 2	98’630137	82’716049	93’333333	93’750000	96’00000
		Clase 3	100’00000	92’857143	100’00000	100’00000	100’00000
		Clase 4	100’00000	93’750000	100’00000	100’00000	100’00000

Media		**entrenamiento**	**test**
Red Inicial	**Global**	**96’96176**	**95’94990**
	Clase 1	98’10001	97’69005
	Clase 2	92’56333	91’57747
	Clase 3	98’53776	89’29762
	Clase 4	100’0000	98’75000
Red Final	**Global**	**98’958304**	**97’512441**
	Clase 1	99’44240	98’93461
	Clase 2	97’13307	92’88590
	Clase 3	100’0000	98’57143
	Clase 4	99’59184	98’75000

	Bien clasificados	Mal clasificados	
Difusos	5’6	5’6	**11’2**
No difusos	1334’8	36’4	**1371’2**
	1340’4	**42**	**1382’4**

Con este método también se consiguen buenos resultados. Mejora para todas las clases, exceptuando la *clase4*. Hay que resaltar que esta clase es la que menos patrones tiene. En el entrenamiento llega a empeorar (*fold2*), aunque en poca medida (aprox. en 0’4 puntos), y en el test se mantiene. Al ser la clase más “pequeña” y la única que empeora, se podría determinar que se desvía de nuestro objetivo (abarcar las clases más minoritarias); pero comparando también el resto de resultados, todos ellos se mejoran, inclusive la *clase3* que también es muy minoritaria, por lo que se alcanza el objetivo de mejorar el porcentaje de aciertos (en modo general).

Respecto a la tabla de relación difusión-clasificación, al igual que en el anterior dominio se recogen menos patrones difusos que patrones mal clasificados. Curiosamente también se produce que el conjunto de patrones difusos está formado en iguales proporciones de patrones bien clasificados y mal clasificados; resaltando que una gran cantidad de los patrones mal clasificados no han sido “incluidos” dentro del cjto. de difusos.

En este método también se muestran los porcentajes de aciertos para cada clase (con el conjunto de entrenamiento) de los individuos que conforman el frente de Pareto resultante en cada *fold*. Como se puede observar, se sigue dando la gran disminución del número de individuos del frente de Pareto resultante.

Frente de Pareto		***clase1***	***clase2***	***clase3***	***clase4***
fold1	*indiv1*	99’4808	98’0707	100	100
	indiv2	99’6885	96’463	100	100
	indiv3	99’5846	96’7846	100	100
fold2	*indiv1*	98’5641	98’0198	100	100
	indiv2	99’6923	93’7294	100	100
	indiv3	98’8718	98’0198	98’1818	100
	indiv4	99’5897	96’0396	100	100
	indiv5	99’4872	97’0297	100	100
fold3	*indiv1*	99’7932	0	0	15’0943
	indiv2	99’3795	97’7346	100	100
	indiv3	99’0693	98’0583	98’1481	100
	indiv4	99’5863	96’4401	100	100
fold4	*indiv1*	99’4845	95’0658	100	100
	indiv2	99’0722	95’7237	100	100
	indiv3	99’1753	95’3947	98’2456	100
fold5	*indiv1*	99’5855	97’411	100	100
	indiv2	98’7565	99’0291	100	100

4.2.5.- Análisis

En comparación con el anterior dominio, en este se dan muy buenos resultados, pues en todos los métodos siempre se logra una mejora en los aciertos, tanto para el conjunto de entrenamiento como para el de test. Para tener una mejor visión general de los resultados obtenidos, se muestra una tabla con los tres métodos y las medias de los porcentajes de aciertos globales; mostrando también la diferencia de porcentaje de la red inicial a la final. El conjunto de entrenamiento ha sido nombrado como *train*.

Red	**Método 1**		**Método 2**			**Método 3**	
	train	***test***	***train1***	***train2***	***test***	***train***	***test***
Inicial	96’9618	95’4990	97’1801	95’3623	96’0069	96’9618	95’9499
Final	99’2037	97’9167	98’7166	99’0580	97’7433	98’9583	97’5124
DIF	**2’2419**	**2’4177**	**1’5365**	**3’6957**	**1’7364**	**1’9965**	**1’5625**

El mejor resultado que se consigue para el conjunto de test es con el *metodo1*, seguido del *metodo2*. Como se indicó en el anterior dominio, la comparación de los conjuntos de entrenamiento es más compleja, pues en el *metodo2* el conjunto está dividido en dos partes, que realizan funciones distintas. Comparando el *metodo1* y el *metodo3*, el que consigue mejores resultados en entrenamiento es el *metodo1*. Por tanto, para este dominio es más efectiva la replicación de los patrones mal clasificados que la replicación de los patrones difusos. Esto seguramente se debe al desequilibrio existente entre las clases.

Si realizamos la comparación con el conjunto *entrenamiento2* del *metodo2*, éste supera al resto de métodos. Esto es destacable, pues este conjunto es el encargado de encauzar el algoritmo evolutivo; por lo que se podría decir que es más importante el conjunto de “evaluación” que el conjunto de entrenamiento en sí de la red. Como se puede también observar, *entrenamiento1* es el conjunto que menos incrementa su tasa de aciertos.

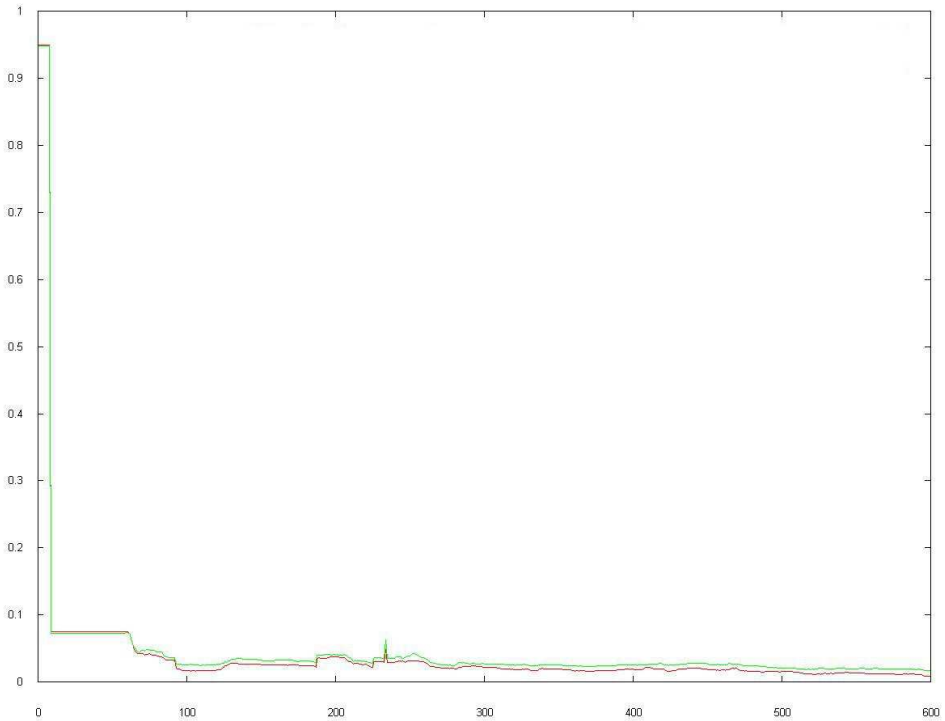
4.3.- Dominio *Thyroides*

Este dominio se corresponde con los datos recogidos del repositorio UCI en el ámbito de la ***Thyroid Disease*** (enfermedad de la tiroides). Este conjunto de datos ha sido suministrado por el Instituto Garavan y J. Ross Quinlan (Sidney, Australia). Está formado por 7.200 instancias que recoge los datos clínicos de un paciente y si padece de hipotiroidismo. Cada instancia recogida se compone de **21 atributos** (15 de ellos con un valor binario y 6 con valor continuo) y puede ser catalogado en **3 clases**. Los atributos se corresponden con datos clínicos del paciente, y se realiza una clasificación en tres clases según el funcionamiento de la tiroides: hipotiroidismo, hipertiroidismo, normal (el 92% de los pacientes). Se denominarán ***clase1*** (166 instancias), ***clase2*** (368 instancias y ***clase2*** (6.666 instancias).

En este dominio no se ha utilizado la validación cruzada, pues el número de datos es muy elevado y el repositorio ofrece de forma separada el conjunto de entrenamiento (3.772 patrones) y el de test (3.428 patrones).

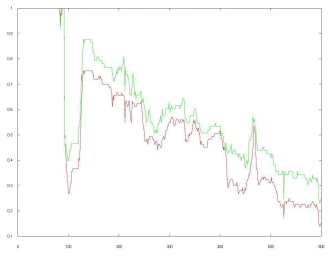
4.3.1.- Elección de la red

Para este dominio el conjunto total de datos (7.200 patrones) es proporcionado por el repositorio en dos conjuntos: **entrenamiento** (**3.772** patrones) y **test** (**3.428**). Se han realizado tres ejecuciones durante 600 iteraciones y con una **razón de aprendizaje de 0'1**, con distinto número de neuronas ocultas: 15, 20 y 25. Se muestra la evolución del error de clasificación global y para cada clase en cada una de arquitecturas entrenadas.

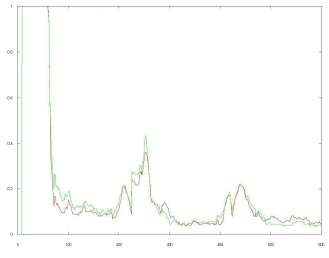


Error de clasificación global

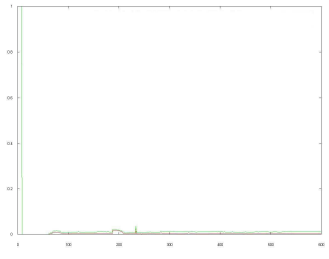
15 neuronas



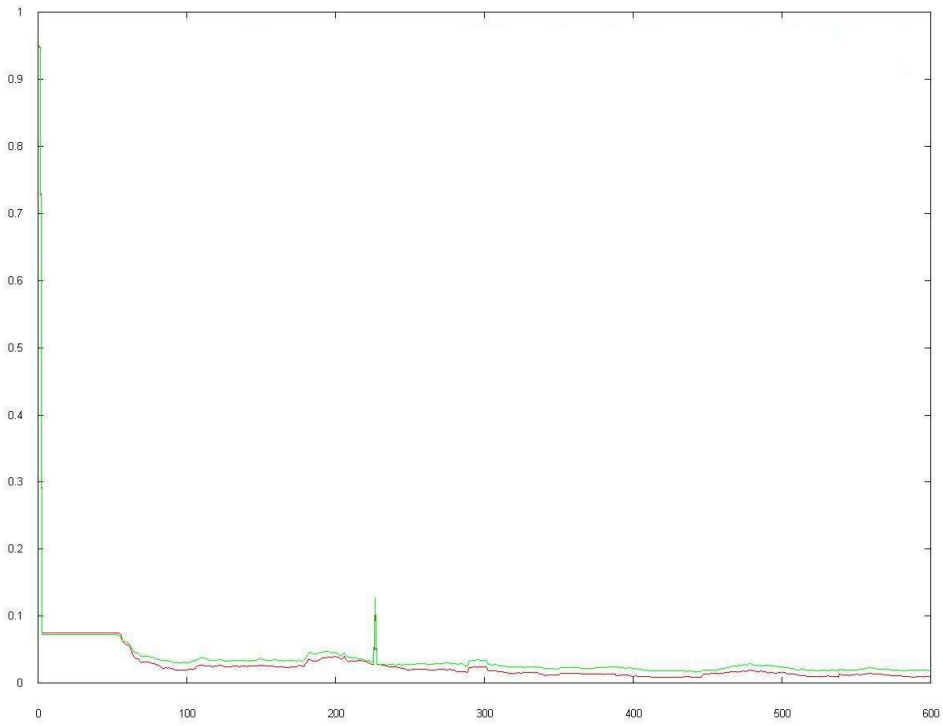
Error clasificación clase1



Error clasificación clase2

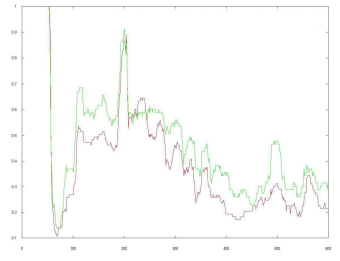


Error clasificación clase3

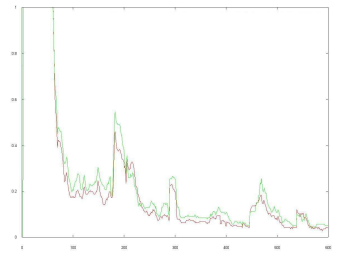


Error de clasificación global

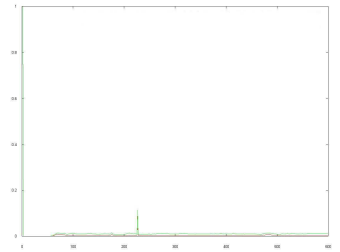
20 neuronas



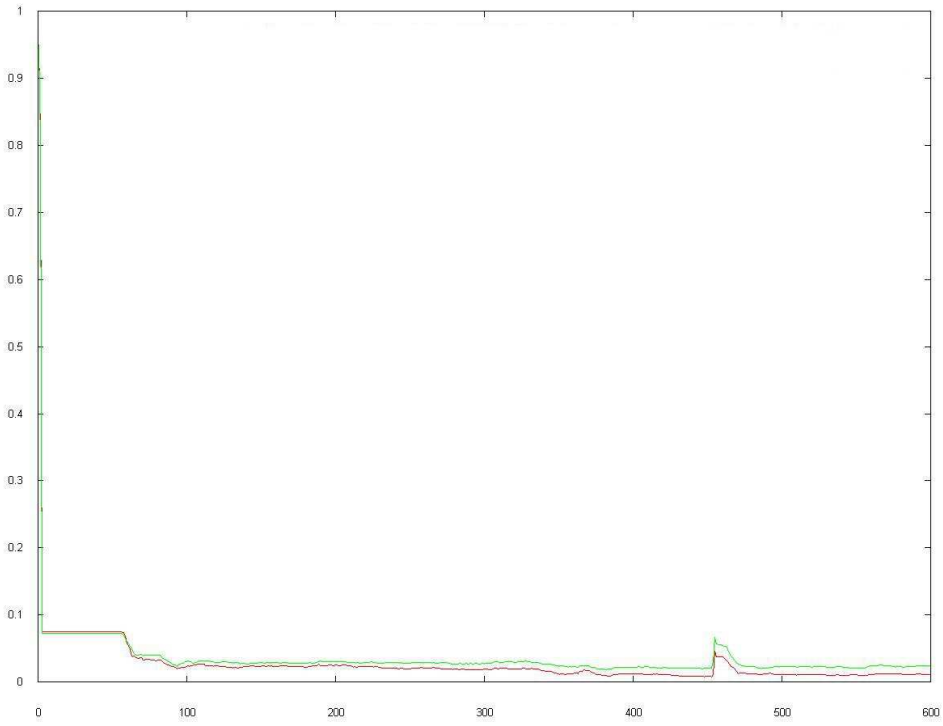
Error clasificación clase1



Error clasificación clase2

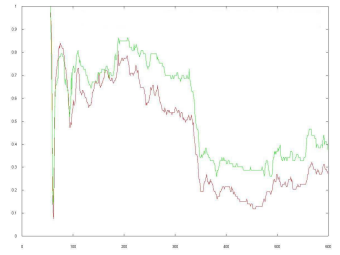


Error clasificación clase3

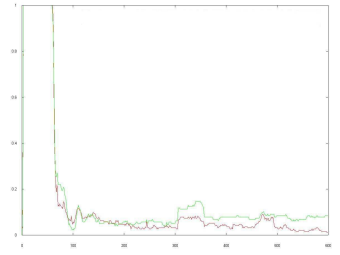


Error de clasificación global

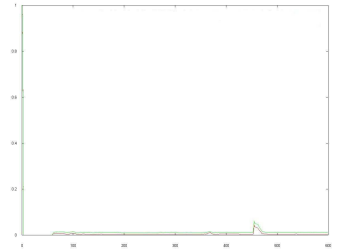
25 neuronas



Error clasificación clase1



Error clasificación clase2



Error clasificación clase3

El comportamiento de las tres redes probadas es muy similar, pues los errores de clasificación global en entrenamiento y test se desarrollan de forma semejante a lo largo de las iteraciones, obteniéndose muy buenos resultados. Si observamos los errores de clasificación para cada una de las clases, se puede comprobar que el error para la *clase3* es mucho menor comparado con el error que se produce en las otras clases. Esto es debido, al igual que en el anterior dominio, a que se trata de un conjunto de datos *desequilibrado*, por lo que también se estudiará si se consigue una mejora en la tasa de aciertos para las clases minoritarias: *clase1* y *clase2*.

Se ha optado por la red con **20 neuronas ocultas**, pues es un número aceptable y el número de atributos es 21, por lo que menos neuronas ocultas podría no ser muy conveniente. La **razón de aprendizaje** se mantiene en **0'1** y se ha optado por realizar el entrenamiento de las redes durante **500 iteraciones**.

4.3.2.- Método 1

En la siguiente tabla se muestran los resultados obtenidos tras aplicar el algoritmo evolutivo: los porcentajes de aciertos globales y para cada clase, tanto para la red inicial como para la red final (el mejor punto del frente de Pareto).

Porcentaje de aciertos		**entrenamiento**	**test**
Red Inicial	**Global**	**98'674443**	**97'666278**
	Clase 1	82'795699	68'493151
	Clase 2	90'575916	88'700565
	Clase 3	99'541284	98'835746
Red Final	**Global**	**99'628844**	**98'249708**
	Clase 1	93'548387	78'082192
	Clase 2	97'905759	93'785311
	Clase 3	99'885321	98'961611

Para el porcentaje de aciertos global, se puede observar que se aumenta la precisión tanto en el conjunto de entrenamiento (en 0'95 puntos) como en el conjunto de test (en 0'58 puntos). Se consiguen unos buenos resultados, pero que no son muy destacables desde el punto de vista global.

El aumento de porcentaje de aciertos más notable se da para la *clase1* y la *clase2*. Para el conjunto de entrenamiento se consigue un aumento de 10'75 para la *clase1* y 7'33 para la *clase2*. Para el conjunto de test se consigue un incremento de 9'59 para la *clase1* y 5'08 para la *clase2*. En cambio para la *clase3* (cuyos patrones suponen más del 92% del total), se produce un aumento mucho menos significativo: 0'34 en el entrenamiento y 0'13 en el test. Estos altos incrementos para las clases minoritarias son unos resultados muy positivos, pues se consigue uno de nuestros objetivos, conseguir que el aprendizaje de la red no se centralice en los patrones de la clase más mayoritaria.

Al igual que en el anterior dominio, no es posible realizar una gráfica legible que muestre la evolución del frente de Pareto durante el algoritmo genético. Por tanto, se muestra una tabla con los mejores individuos que resultan en el frente de Pareto al finalizar el algoritmo, indicando los porcentajes de acierto para cada clase del dominio.

Frente de Pareto	***clase1***	***clase2***	***clase3***
individuo1	93’5484	97’9058	99’8853
individuo2	87’0968	97’9058	99’9427
individuo3	53’7634	99’4764	99’8567
individuo4	79’5699	98’9529	99’8853
individuo5	86’0215	100	99’828
individuo6	88’172	98’9529	99’8567
individuo7	88’172	99’4764	99’7706
individuo8	84’9462	98’4293	99’914
individuo9	91’3978	98’4293	99’8567
individuo10	89’2473	98’9529	99’828
individuo11	87’0968	99’4764	99’828
individuo12	92’4731	98’4293	99’7993

4.3.3.- Método 2

En este método se divide el cjto. de entrenamiento original (3.772 patrones) en dos partes: el 80% es *entrenamiento1* (3.018 patrones) y el 20% restante es *entrenamiento2* (754 patrones). Se seguirá el procedimiento descrito en el ***Apartado 3.3.1***, con el conjunto de *entrenamiento1* se entrenará la red con los patrones mal clasificados replicados, y con el conjunto de *entrenamiento2* se evaluará el *fitness* del individuo.

Porcentaje de aciertos		**entrenamiento1**	**entrenamiento2**	**test**
Red Inicial	**Global**	**94’400265**	**94’694960**	**93’553092**
	Clase 1	44’303797	57’142857	36’986301
	Clase 2	20’000000	17’073171	14’124294
	Clase 3	99’820724	100’00000	99’276274
Red Final	**Global**	**99’204771**	**99’204244**	**97’782964**
	Clase 1	86’075949	100’00000	72’602740
	Clase 2	100’00000	95’121951	93’785311
	Clase 3	99’533883	99’427754	98’584015

Para todos los conjuntos utilizados se produce un aumento en el porcentaje de aciertos global: 4’80 puntos en *entrenamiento1*, 4’51 puntos en *entrenamiento2* y 4’22 puntos en test. Sin embargo lo más significativo de estos resultados se produce en el incremento de aciertos en cada una de las clases.

En la red inicial se dan unos resultados pésimos para la *clase1* y la *clase2*; mientras que con la *clase3* (la más mayoritaria) se producen unos buenos porcentajes de aciertos.

Lo más llamativo es el gran incremento de aciertos que se producen en las clases minoritarias con la red final. Para la *clase1* aumenta en 41'77 para *entrenamiento1*, 42'86 para *entrenamiento2* y 34'62 para test. Para la *clase2* se produce un incremento de 80 puntos para *entrenamiento1*, 78'05 para *entrenamiento2* y 79'66 para test. Sin embargo, en la *clase3* se produce una disminución en los porcentajes de aciertos: 0'28 puntos para *entrenamiento1*, 0'57 para *entrenamiento2* y 0'69 para test.

Por tanto los resultados son muy favorables, pues hay un gran aumento para las clases minoritarias, aunque la clase mayoritaria tenga pérdidas; además los porcentajes globales también aumentan.

A continuación se muestra una tabla con los individuos que conforman el frente de Pareto al finalizar el algoritmo, indicando el porcentaje de aciertos que obtienen los distintos individuos para cada una de las clases. Estos resultados son los que se producen al evaluar la red con el conjunto de *entrenamiento2*.

Frente de Pareto	***clase1***	***clase2***	***clase3***
individuo1	100	95'122	99'4278
individuo2	71'4286	85'3659	99'8569
individuo3	57'1429	97'561	99'5708
individuo4	85'7143	95'122	99'5708
individuo5	78'5714	95'122	99'7139
individuo6	85'7143	92'6829	99'7139
individuo7	92'8571	90'2439	99'7139
Individuo8	92'8571	97'561	99'4278

4.3.4.- Método 3

En este método se replican los patrones difusos en vez de los mal clasificados (***Apartado 3.3.2***). A continuación, se muestran los porcentajes de acierto obtenidos para la red inicial y final para los conjuntos de entrenamiento y de test, diferenciando también cada una de las clases. También se incluye una tabla que relaciona los dos criterios utilizados para la replicación de los patrones: *difusión* y *clasificación*; indicando el número de patrones que hay de cada tipo en el conjunto de entrenamiento original.

Porcentaje de aciertos		**entrenamiento**	**test**
Red Inicial	**Global**	**98'674443**	**97'666278**
	Clase 1	82'795699	68'493151
	Clase 2	90'575916	88'700565
	Clase 3	99'541284	98'835746
Red Final	**Global**	**99'469777**	**98'220537**
	Clase 1	87'096774	67'123288
	Clase 2	98'429319	94'915254
	Clase 3	99'856651	99'118943

Al igual que con los anteriores métodos, se produce una mejoría en los porcentajes de aciertos globales: 0’795 puntos con el conjunto de entrenamiento y 0’554 con el de test. Pero las grandes diferencias se producen en los porcentajes de aciertos por clase. En la *clase1*, la que menos patrones tiene, tiene un aumento de 4’301 puntos con el conjunto de entrenamiento, pero sin embargo se produce una disminución en 1’370 puntos con el conjunto de test. En la *clase2* el aumento es mayor y se produce para los dos conjuntos: en entrenamiento de un 7’853 y en test de 6’215. Y para la clase más mayoritaria (*clase3*) se produce un incremento de 0’315 puntos en entrenamiento y 0’283 en test.

En general son unos resultados bastante buenos, aunque hay que destacar que la *clase1* en el conjunto de test tiene un porcentaje de aciertos más o menos admisible en la red inicial, pero llega incluso a empeorar en la red final.

	Bien clasificados	**Mal clasificados**	
Difusos	9	14	**23**
No difusos	3713	36	**3749**
	3722	**50**	**3772**

Como se ha producido en los anteriores dominios, el número de patrones difusos que se recogen es menor al número de patrones mal clasificados. Sin embargo, en esta ocasión el conjunto de patrones difusos está formado por más patrones mal clasificados que bien clasificados. Hay que resaltar, los pocos patrones mal clasificados, en comparación con el total de patrones, que se obtienen en la red inicial. De todos modos, también se siguen sin recoger como patrones difusos muchos de los patrones mal clasificados.

El frente de Pareto resultante tras la ejecución del algoritmo evolutivo mediante este método, es el representado en la próxima tabla, en la que se indican los porcentajes de aciertos por clase para cada uno de los individuos del frente.

Frente de Pareto	***clase1***	***clase2***	***clase3***
individuo1	78'4946	93'7173	99'914
individuo2	91'3978	97'9058	99'4266
individuo3	89'2473	96'8586	99'7706
individuo4	87'0968	98'4293	99'8567
individuo5	83'871	98'9529	99'8567
individuo6	89'2473	97'3822	99'5986
individuo7	90'3226	96'3351	99'8567
individuo8	88'172	97'3822	99'828
individuo9	34'4086	94'7644	99'914
individuo10	84'9462	98'9529	99'7706
individuo11	88'172	98'4293	99'7133
individuo12	82'7957	97'9058	99'8853
individuo13	80'6452	99'4764	99'828

4.3.5.- Análisis

Para los tres métodos se consigue una mejora en los porcentajes de aciertos en la clasificación global. Para tener una visión general y contrastar los distintos métodos, se muestra a continuación una tabla con los distintos porcentajes de aciertos globales para cada uno de los conjuntos de datos. El conjunto de entrenamiento ha sido nombrado como *train*, y son mostrados 4 decimales para mejorar la legibilidad.

Red	**Método 1**		**Método 2**			**Método 3**	
	train	***test***	***train1***	***train2***	***test***	***train***	***test***
Inicial	98’6744	97’6663	94’4003	94’6950	93’5531	98’6744	97’6663
Final	99’6288	98’2497	99’2048	99’2042	97’7830	99’4698	98’2205
DIF	**0’9544**	**0’5834**	**4’8045**	**4’5092**	**4’2299**	**0’7954**	**0’5542**

La mayor mejora se produce en el *metodo2*, pero aún así, el resultado de la red final es peor que en los otros dos métodos. Esto es debido a que los resultados iniciales son peores, pues el conjunto de datos de entrenamiento de la red es menor (*train1*), por lo que aunque se logré un gran incremento tras el algoritmo evolutivo, no se alcanzan los resultados de la red final con el método original (*método1*).

Si comparamos el *metodo1* y el *metodo3*, la mayor mejora se produce en el *método1*. Esto se ha producido también en los anteriores dominios, por lo que como también se comentó: la replicación de patrones mal clasificados es mejor que la replicación de los patrones difusos.

De todos modos, lo más reseñable de los resultados de este dominio es el gran incremento que se produce, entre la red inicial y la red final, en las clases minoritarias (*clase1* y *clase2*), aunque en la clase más mayoritaria (*clase3*) se llegue en ocasiones a producir una disminución en su porcentaje de aciertos. Esto ya ha sido comentado en más detalle en las diferentes secciones en las que se ha dividido, según el método utilizado, este dominio.

Por tanto, se obtienen unos resultados muy buenos, principalmente en el *metodo1*, pues se consigue una red que generaliza mejor aquellos patrones pertenecientes a las clases minoritarias, consiguiendo una mayor cobertura de todos los patrones independientemente de la clase a la que pertenezcan.

4.4.- Dominio *Balance-Scale*

Este dominio se corresponde con el dominio ***Balance-Scale*** (balanza) del repositorio UCI. Está compuesto de 645 instancias que recogen resultados experimentales realizados en una balanza. Cada instancia (patrón) está representado por **4 atributos** y puede ser catalogado entre **3 clases**. Los atributos indican los valores de la balanza para el experimento realizado, siendo estos:

- Peso del lado izquierdo: *1, 2, 3, 4, 5*
- Longitud del brazo izquierdo: *1, 2, 3, 4, 5*
- Peso lado derecho: *1, 2, 3, 4, 5*
- Longitud del brazo derecho: *1, 2, 3, 4, 5*

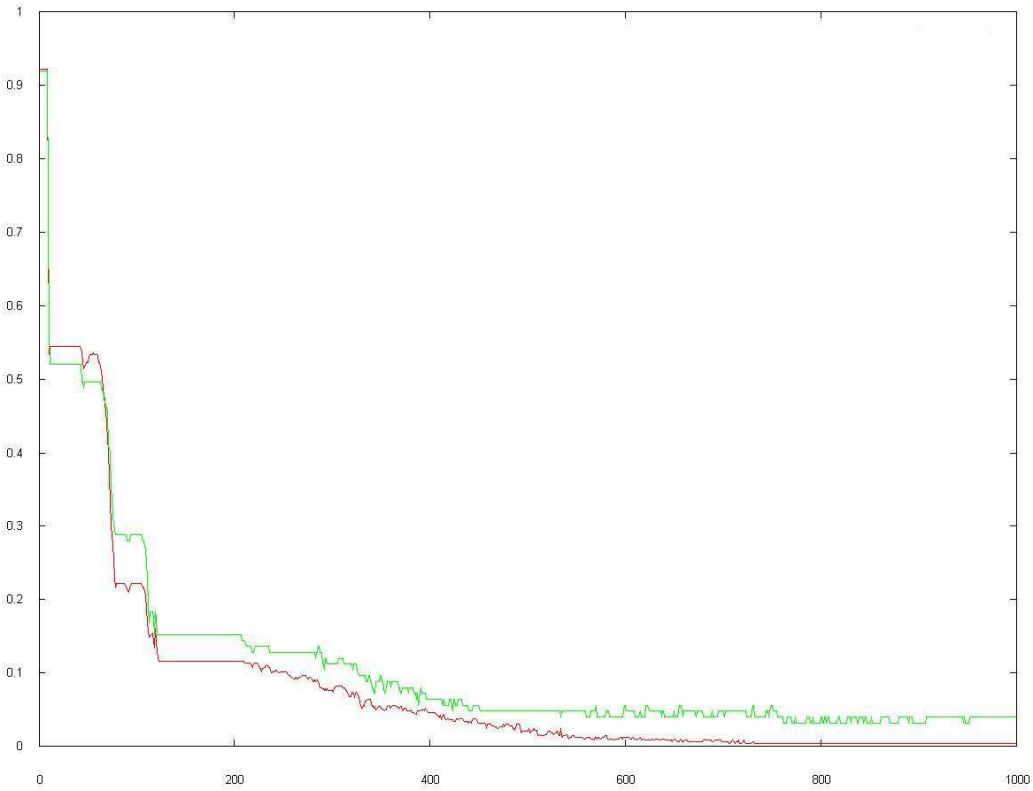
La clase indicará la posición de la aguja de la balanza: hacia la izquierda, hacia la derecha o está balanceada. Para conocer la posición de la aguja (la clase) se comparan las siguientes ecuaciones: [*peso_izda*long_izda*] y [*peso_dcha*long_dcha*)]. Si la ecuación del lado izquierdo es mayor que la del lado derecho, la aguja de la balanza se posiciona hacia la izquierda; y viceversa, si el valor de la ecuación de la derecha es mayor que la de la izquierda, la balanza se inclinará hacia la derecha. En el caso de que ambos valores sean exactamente iguales, la balanza estará equilibrada. La distribución de las instancias en cada clase es la siguiente:

- Inclinada a la izquierda: *clase1* (288 patrones)
- Balanceada: *clase2* (49 patrones)
- Inclinada a la derecha: *clase3* (288 patrones)

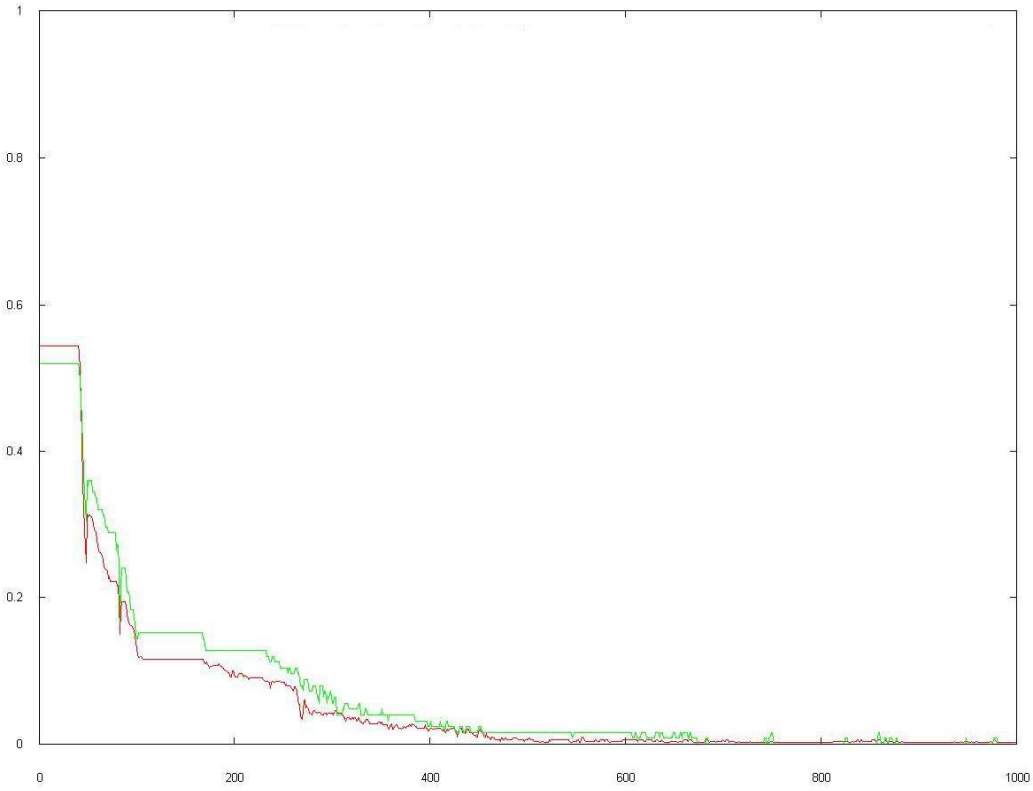
4.4.1.- Elección de la red

En primer lugar el conjunto de datos se ha dividido en dos partes: el **80%** para **entrenamiento** (500 patrones) y el **20%** restante para **test** (125 patrones). Tras ello se han realizado tres ejecuciones con una **razón de aprendizaje de 0'1**, para tres redes de neuronas con distinto número de neuronas ocultas: **15**, **20** y **25**.

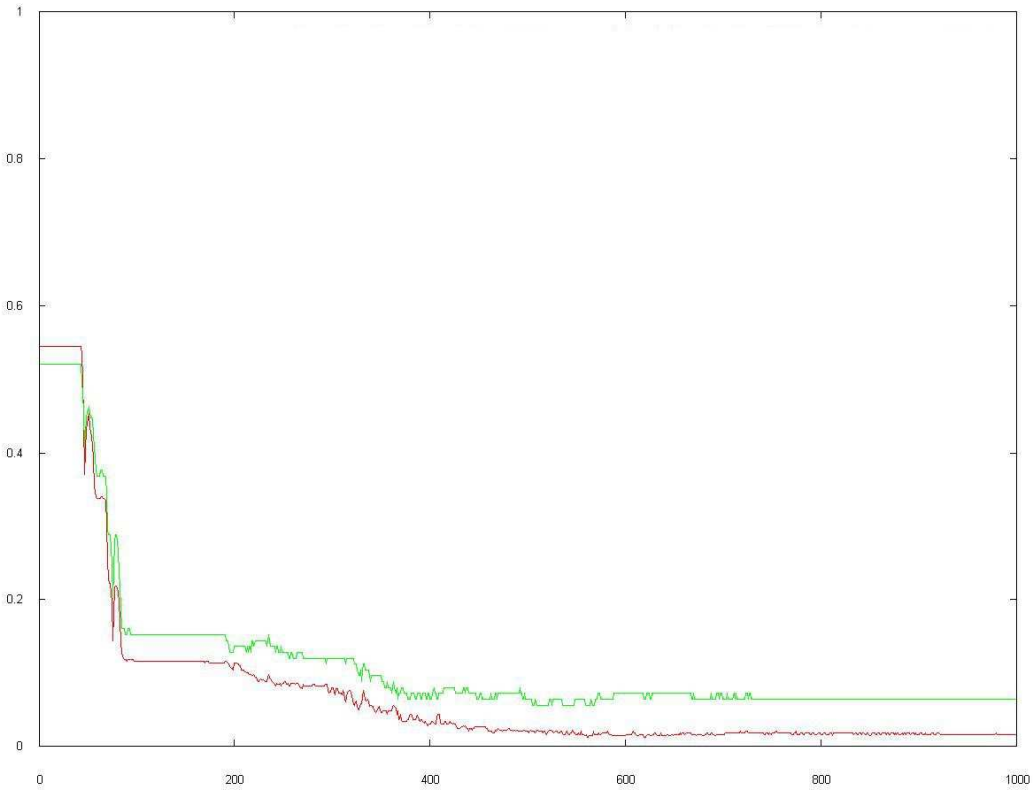
A continuación se muestran las tres gráficas correspondientes a las distintas redes de neuronas seleccionadas. En ellas se muestra el error de clasificación para el entrenamiento (rojo) y para el test (verde) durante **1.000 iteraciones**.



Red con 15 neuronas ocultas



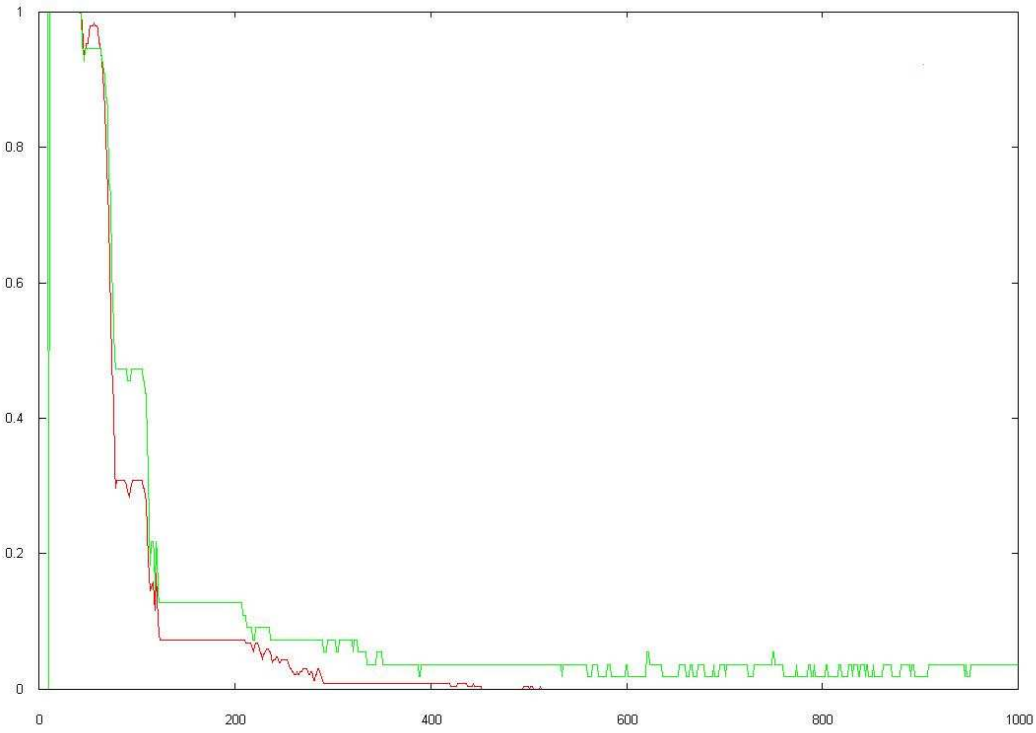
Red con 20 neuronas ocultas



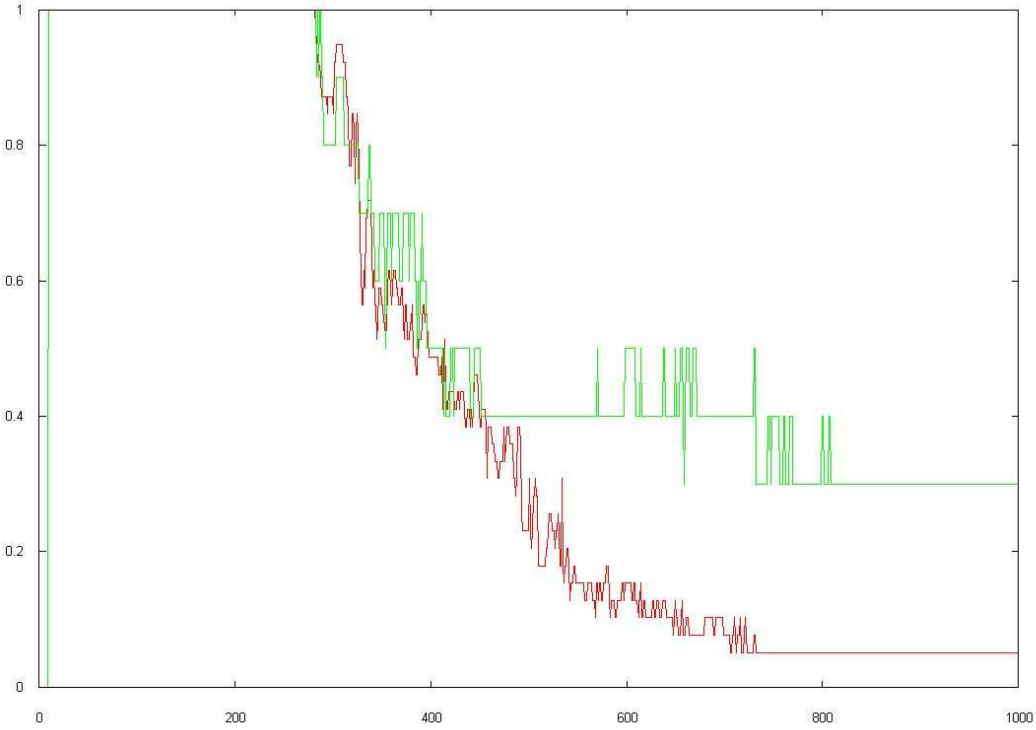
Red con 25 neuronas ocultas

Observando el error de entrenamiento (rojo), para las redes de 15 y 20 neuronas ocultas es muy similar; pero para la red de 25 neuronas ocultas es algo mayor. En función del error de test, el mayor error que se produce es en la red de 25 de neuronas ocultas. Por tanto se optará por una de las otras dos redes de neuronas. Se ha escogido la red de 15 neuronas ocultas, al ser la que menor coste computacional supone, aunque a lo largo de las iteraciones el error de test sea algo mayor que la red con 20 neuronas ocultas.

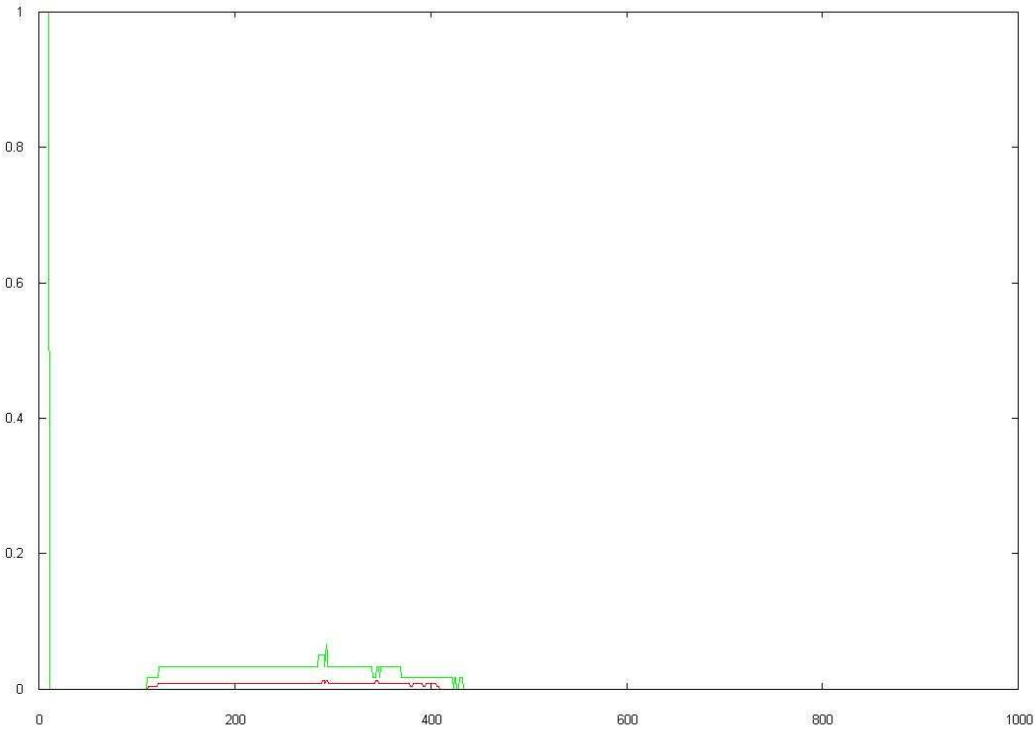
A continuación se muestran las gráficas de los errores de clasificación en entrenamiento y test que se produce para cada una de las clases. Se trata del entrenamiento de la misma red que se ha mostrado anteriormente.



Error de clasificación de la clase1



Error de clasificación de la clase2



Error de clasificación de la clase3

En base a la gráfica de error de clasificación global de la red de 15 neuronas, se puede observar que el resultado es bastante bueno. Pero a partir de los errores de clasificación por clase, se observa que para las clases 1 y 3 se obtienen muy buenas clasificaciones, mientras que en la *clase2* se producen unos peores resultados. Esta clase es la más pequeña, pues contiene 49 patrones, mientras que las otras dos tienen 288 patrones cada una.

Por tanto se trata de un dominio con datos *desequilibrados*, al igual que otros de los dominios contemplados anteriormente, y esto produce que la clasificación en la clase minoritaria sea peor, pues la red se concentra en la clasificación del mayor número de patrones y por tanto de las clases más grandes. Por tanto, el estudio se centrará en comprobar si se logra una mejoría en la *clase2* con la mínima perdida de precisión en la clasificación global.

4.4.2.- Método 1

A continuación se muestran los resultados obtenidos tras aplicar la validación cruzada con el algoritmo evolutivo mediante el *método1* (***Apartado 3.2***). Se muestran los porcentajes de aciertos globales y para cada clase, para la red inicial y para la red final (el mejor punto del frente de Pareto). También se ha realizado la media de los porcentajes de aciertos que se obtienen entre todos los *folds*.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento	**Red Inicial**	**Global**	94’000000	95’800000	97’400000	98’400000	98’800000
		Clase 1	100’00000	100’00000	100’00000	99’555555	100’00000
		Clase 2	16’666667	43’243243	69’767442	85’365854	84’615385
		Clase 3	100’00000	100’00000	100’00000	99’572650	100’00000
	Red Final	**Global**	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 1	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 2	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 3	100’00000	100’00000	100’00000	100’00000	100’00000
test	**Red Inicial**	**Global**	92’000000	91’200000	96’800000	96’000000	94’400000
		Clase 1	100’00000	100’00000	98’181818	96’825397	96’363636
		Clase 2	30’769231	8’333333	50’000000	62’000000	50’000000
		Clase 3	98’039216	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	100’00000	96’800000	99’200000	98’400000	100’00000
		Clase 1	100’00000	100’00000	100’00000	98’412698	100’00000
		Clase 2	100’00000	83’333333	100’00000	87’500000	100’00000
		Clase 3	100’00000	96’610169	98’437500	100’00000	100’00000

Media		**entrenamiento**	**test**
Red Inicial	**Global**	**96’880000**	**94’080000**
	Clase 1	99’911111	98’274170
	Clase 2	59’931718	40’220513
	Clase 3	99’914530	99’607843
Red Final	**Global**	**100’00000**	**98’880000**
	Clase 1	100’00000	99’682540
	Clase 2	100’00000	94’166667
	Clase 3	100’00000	99’009534

En todos los *folds* se alcanza un 100% de aciertos con el conjunto de entrenamiento, tanto para la clasificación global como para cada una de las clases. Respecto al conjunto de test también se logran muy buenos resultados, obteniéndose un 100% de aciertos en el *fold1* y en el *fold5*, y en el resto se dan porcentajes mayores al 95%.

Lo más resaltable se produce en la tasa de aciertos para la *clase2*, que es precisamente la más minoritaria. Para esta clase se produce un gran incremento en todos los *folds*, tanto con el conjunto de entrenamiento como con el de test. Los incrementos más destacables se dan en el *fold1* con el conjunto de entrenamiento, que aumenta en 83’33 puntos (de un 16’67% a un 100%); y en el *fold2* con el conjunto de test que aumenta en 75 puntos, de un 8’33% a un 83’33%.

Las únicas disminuciones que se dan entre la red inicial y la final se producen en dos grupos, ambos se producen con el conjunto de test y sobre el porcentaje de aciertos de la *clase3*. Estas dos disminuciones se dan lugar en el *fold2* y en el *fold3*.

Por tanto se consiguen unos resultados muy buenos. Tras las medias realizadas entre todos los *folds*, se puede observar que se consigue un aumento de 3'12 puntos con el conjunto de entrenamiento y de 4'8 puntos con el conjunto de test, en base al porcentaje de aciertos global.

Como ocurría en los últimos dominios estudiados, no se puede realizar una representación gráfica de la evolución de los frentes, pues este dominio tiene 3 clases. El frente de Pareto que resulta del algoritmo evolutivo, en todos los *folds*, está formado por individuos que tienen un porcentaje de aciertos del 100% para todas las clases y, por consiguiente también en el porcentaje de aciertos global.

4.4.3.- Método 2

En este método el conjunto de entrenamiento original (500 patrones) es dividido en dos partes: el 80% es *entranamiento1* (400 patrones) y el 20% es *entranamiento2* (100 patrones). La primera de ellas se utilizará para entrenar la red y la segunda para evaluar la red en el *fitness* y en la elección del mejor individuo del frente de Pareto.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento1	**Red Inicial**	**Global**	97'250000	97'000000	99'000000	97'500000	98'000000
		Clase 1	98'369565	98'342541	99'476440	98'901099	98'888888
		Clase 2	69'230769	68'965517	91'428571	78'378378	80'645161
		Clase 3	100'00000	100'00000	100'00000	100'00000	100'00000
	Red Final	**Global**	99'500000	98'000000	97'250000	99'500000	99'250000
		Clase 1	99'456522	99'447514	100'00000	100'00000	100'00000
		Clase 2	96'153846	79'310345	68'571429	97'297297	93'548387
		Clase 3	100'00000	99'473684	100'00000	99'447514	99'470899
entrenamiento2	**Red Inicial**	**Global**	93'000000	96'000000	96'000000	97'000000	93'000000
		Clase 1	97'674419	98'113208	100'00000	100'00000	98'113208
		Clase 2	50'000000	62'500000	62'500000	50'000000	37'500000
		Clase 3	97'872340	100'00000	98'000000	98'113208	97'435897
	Red Final	**Global**	99'000000	98'000000	99'000000	100'00000	97'000000
		Clase1	100'00000	100'00000	100'00000	100'00000	96'226415
		Clase 2	100'00000	75'000000	87'500000	100'00000	87'500000
		Clase 3	97'872340	100'00000	100'00000	100'00000	100'00000
test	**Red Inicial**	**Global**	96'000000	93'600000	96'800000	94'400000	91'200000
		Clase 1	98'360656	100'00000	98'181818	95'238095	94'545455
		Clase 2	76'923077	33'333333	66'666667	50'000000	40'000000
		Clase 3	98'039216	100'00000	98'437500	100'00000	96'666667
	Red Final	**Global**	98'400000	96'000000	96'800000	96'000000	97'600000
		Clase 1	98'360656	100'00000	100'00000	95'238095	94'545455
		Clase 2	100'00000	66'666667	66'666667	75'000000	100'00000
		Clase 3	98'039216	98'305085	96'875000	100'00000	100'00000

Media		entrenamiento1	entrenamiento2	test
Red Inicial	**Global**	**97'750000**	**95'000000**	**94'400000**
	Clase 1	98'795707	98'780167	97'265205
	Clase 2	77'729679	52'500000	53'384615
	Clase 3	100'00000	98'284289	98'628677
Red Final	**Global**	**98'700000**	**98'600000**	**96'960000**
	Clase 1	98'780807	99'245283	97'628841
	Clase 2	86'976261	90'000000	81'666667
	Clase 3	99'678419	99'574468	98'643860

Para todos los conjuntos utilizados se produce un aumento en el porcentaje de aciertos global en todos los *folds*, y en base a las medias realizadas se observa un aumento de 0'95 puntos para *entrenamiento1*, de 3'6 puntos para *entrenamiento2* y 2'56 en el conjunto de test. Como ocurre con el resto de dominios, la mayor mejora se produce con el conjunto de *entrenamiento2* y el conjunto de test no logra tan buenos resultados como con el *metodo1*. El gran incremento de la tasa de aciertos con el conjunto de *entrenamiento2* se debe a que este conjunto es el que utiliza el algoritmo evolutivo como función de *fitness*.

El incremento de aciertos más significativo sigue siendo con la *clase2*, que es la más minoritaria, sobre todo con el conjunto de *entrenamiento2*. En las otras clases, las más mayoritarias, se produce en algunos casos un decremento del número de aciertos, pero no llegan a ser muy significativos. Por tanto, sigue siendo un buen método para aumentar el número de aciertos, pues además de mejorar la clasificación de los patrones más minoritarios, el incremento también se produce en el porcentaje de aciertos global.

En la siguiente tabla se muestra el frente de Pareto resultante tras realizar el algoritmo evolutivo, mostrándose el porcentaje de aciertos para cada clase que se originan con el conjunto de *entrenamiento2*.

Frente de Pareto		***clase1***	***clase2***	***clase3***
fold1	*indiv1*	100	100	97'8723
	indiv2	97'6744	100	100
	indiv3	100	80	100
fold2	*indiv1*	100	75	100
fold3	*indiv1*	100	87'5	100
fold4	*indiv1*	100	100	100
fold5	*indiv1*	96'2264	87'5	100
	indiv2	100	62'5	97'4359
	indiv3	98'1132	75	100

4.4.4.- Método 3

En este método se replican los patrones difusos en vez de los mal clasificados, tal y como se ha especificado en el ***Apartado 3.3.2***. A continuación se muestran los porcentajes de acierto obtenidos para la red inicial y final en cada uno de los *folds* y para los conjuntos de entrenamiento y de test, diferenciando también cada una de las clases. Asimismo, se muestran las medias de los porcentajes obtenidos entre todos los *folds*. También se indica una tabla que relaciona los dos criterios utilizados para la replicación de los patrones: *difusión* y *clasificación*; que resulta de las medias de dichos valores de cada *fold*.

Porcentaje de aciertos			***fold 1***	***fold 2***	***fold 3***	***fold 4***	***fold 5***
entrenamiento	**Red Inicial**	**Global**	94’000000	95’800000	97’400000	98’400000	98’800000
		Clase 1	100’00000	100’00000	100’00000	99’555556	100’00000
		Clase 2	16’666667	43’243243	69’767442	85’365854	84’615385
		Clase 3	100’00000	100’00000	100’00000	99’572650	100’00000
	Red Final	**Global**	100’00000	100’00000	100’00000	100’00000	99’800000
		Clase 1	100’00000	100’00000	100’00000	100’00000	100’00000
		Clase 2	100’00000	100’00000	100’00000	100’00000	97’435897
		Clase 3	100’00000	100’00000	100’00000	100’00000	100’00000
test	**Red Inicial**	**Global**	92’000000	91’200000	96’800000	96’000000	94’400000
		Clase 1	100’00000	100’00000	98’181818	96’825397	96’363636
		Clase 2	30’769231	8’333333	50’000000	62’500000	50’000000
		Clase 3	98’039216	100’00000	100’00000	100’00000	100’00000
	Red Final	**Global**	98’400000	97’600000	97’600000	96’800000	97’600000
		Clase 1	98’360656	100’00000	98’181818	96’825397	96’363636
		Clase 2	100’00000	83’333333	100’00000	87’500000	90’000000
		Clase 3	98’039216	98’305085	96’875000	98’148148	100’00000

Media		**entrenamiento**	**test**
Red Inicial	**Global**	**96’880000**	**94’080000**
	Clase 1	99’911111	98’274170
	Clase 2	59’931718	40’320513
	Clase 3	99’914530	99’607843
Red Final	**Global**	**99’960000**	**97’600000**
	Clase 1	100’00000	97’946301
	Clase 2	99’487179	92’166667
	Clase 3	100’00000	98’273490

Mediante este método también se logran muy buenos resultados. Con el conjunto de entrenamiento se consigue un incremento en el porcentaje de aciertos para todas las clases, especialmente en la *clase2*, cuyo aumento es mucho más significativo (unos 40 puntos), como ocurría en los anteriores métodos. Con el conjunto de test se produce una leve disminución de aciertos para la *clase1* y *clase3*, mientras que se produce una gran mejora en la tasa de aciertos en la *clase2* (unos 52 puntos). Por tanto, se trata de un buen método pues mejora sustancialmente la tasa de aciertos de la clase minoritaria, aumentando también el porcentaje de aciertos global, aunque suponga una leve pérdida de aciertos para el resto de clases.

	Bien clasificados	Mal clasificados	
Difusos	8’6	12	**20’6**
No difusos	475’8	3’6	**479’4**
	484’4	**15’6**	**500**

En este dominio, al contrario que en los anteriores, se recogen más patrones difusos que patrones mal clasificados. Además, la mayoría de los patrones difusos recogidos son a su vez mal clasificados, quedando muy pocos patrones mal clasificados no recogidos como difusos (un 23%). Esto tampoco se ha producido en los anteriores dominios, pues la mayoría de los patrones mal clasificados no eran recogidos como difusos. Esto seguramente sea la razón por la que en este dominio al utilizar el *método3* los resultados obtenidos son mejores que los adquiridos en los otros dominios.

A continuación se muestra una tabla con los individuos que conforman el frente de Pareto para cada uno de los *folds*. Se muestra el porcentaje de aciertos para cada clase obtenido evaluando con el conjunto de entrenamiento, de la red asociada a cada individuo. En este caso, los frentes se componen de un único punto en la mayor parte de los *folds*, similar al *metodo1* en el que sólo había un único individuo y que conseguía siempre 100% de aciertos.

Frente de Pareto		***clase1***	***clase2***	***clase3***
fold1	*indiv1*	100	100	100
fold2	*indiv1*	100	100	100
fold3	*indiv1*	100	100	100
fold4	*indiv1*	100	100	100
fold5	*indiv1*	100	97’4359	100
	indiv2	100	100	99’5614

4.4.5.- Análisis

Aunque este dominio tenga un número reducido de datos (625 patrones), se han conseguido unos resultados muy satisfactorios. En todos los métodos se ha logrado mejorar la tasa de aciertos tanto para el conjunto de entrenamiento como para el de test. A continuación, se muestra una tabla con los resultados obtenidos para cada método, mostrando también la mejora obtenida.

Red	**Método 1**		**Método 2**			**Método 3**	
	train	***test***	***train1***	***train2***	***test***	***train***	***test***
Inicial	96’8800	94’0800	97’7500	95’0000	94’4000	96’8800	94’0800
Final	100’000	98’8800	98’7000	98’6000	96’9600	99’9600	97’6000
DIF	**3’12**	**4’80**	**0’95**	**3’60**	**2’56**	**3’08**	**3’52**

El mejor resultado que se consigue para el conjunto de test es con el *metodo1*, seguido del *metodo3*. En este dominio, el *metodo3* (replicación de los patrones difusos en vez de los patrones mal clasificados) no ha empeorado tanto los resultados en comparación con los resultados obtenidos con el *metodo1*, algo que en los anteriores dominios se producía en mayor escala. Como en el resto de dominios, en este la elección de un conjunto de validación (conjunto de *entrenamiento2* utilizado en *metodo2*) empeora los resultados obtenidos con el método inicial (*metodo1*).

El resultado más significativo para realizar una evaluación, es la tasa de aciertos obtenida con el conjunto de test, pues este conjunto es totalmente ajeno al algoritmo. Por tanto, el mejor método para este dominio es *metodo1*, aunque en el resto de métodos se consigan también buenos resultados.

5.- CONCLUSIONES Y FUTUROS TRABAJOS

Durante el anterior apartado de ***Experimentación***, ya se han ido analizando los resultados obtenidos y se han plasmado las primeras deducciones realizadas para cada uno de los dominios y métodos utilizados. En este apartado se recogerán las conclusiones generales del proyecto, pero para conocer con más detalle los resultados específicos, se recomienda acudir a la sección anterior.

Con el fin de comparar los resultados obtenidos en todos los dominios estudiados y para cada uno de los métodos utilizados, se muestra a continuación una tabla resumen de los porcentajes de aciertos globales para los conjuntos de entrenamiento y de test.

		Método 1		Método 2			Método 3	
		Entrenamiento	Test	Entrenamiento 1	Entrenamiento 2	Test	Entrenamiento	Test
BUPA	Red Inicial	76,44928	73,33333	76,10860	76,00000	73,04348	76,44928	73,33333
	Mejor Individuo	80,36232	72,46379	73,12217	86,18182	66,95652	80,86957	69,27536
Car	Red Inicial	96,96176	95,94990	97,18007	95,36232	96,00687	96,96176	95,94990
	Mejor Individuo	99,20374	97,91673	98,71658	99,05797	97,74332	98,95830	97,51244
Thyroides	Red Inicial	98,67444	97,66628	94,40027	94,69496	93,55309	98,67444	97,66628
	Mejor Individuo	99,62884	98,24971	99,20477	99,20424	97,78296	99,46978	98,22054
Balance-Scale	Red Inicial	96,88000	94,08000	97,75000	95,00000	94,40000	96,88000	94,08000
	Mejor Individuo	100,00000	98,88000	98,70000	98,60000	96,96000	99,96000	97,60000

En cuanto a los dominios, en el dominio ***BUPA* no se consiguen mejorar los resultados**. La principal causa de esto es el poco número de patrones de los que se dispone (345). Esto causa que la red tenga un **sobreaprendizaje**, lo que se puede apreciar al aumentar el porcentaje de aciertos en el conjunto de entrenamiento, mientras que en el conjunto de test disminuye. En el resto de dominios se consiguen mejores resultados, pues se dispone de un mayor número de datos. También hay que señalar, que en *BUPA* las clases están bastante equilibradas, mientras que en el resto de dominios los datos están desequilibrados.

En cuanto a los métodos utilizados, los valoraremos respecto a los dominios de ***Car***, ***Thyroides*** y ***Balance-Scale***, pues los resultados obtenidos en *BUPA* no son representativos. En cuanto a los conjuntos de datos utilizados, nos fijaremos especialmente en el de **test**, pues es un conjunto totalmente ajeno al algoritmo y nos proporciona una valoración más objetiva para evaluar el sistema.

Respecto al ***metodo1***, se mejoran los resultados tanto en entrenamiento como en test. Aumentan los porcentajes de aciertos para todas las clases, siendo más significativo el aumento en las clases minoritarias (ver tablas del ***Apartado 4.2.2*** y ***Apartado 4.3.2***). Sólo hay un caso en el que se empeora el porcentaje de aciertos, que es en la *clase3* con el conjunto de test en dominio de *Balance-Scale* (ver tablas del ***Apartado 4.4.2***), pero es poco significativo, y más teniendo en cuenta que la clase minoritaria en este dominio (la *clase2*) aumenta su porcentaje de aciertos en gran medida.

Con esto, logramos el objetivo de, en los dominios desequilibrados, aumentar el porcentaje de aciertos en las clases minoritarias sin empeorar en gran medida los aciertos para las clases mayoritarias, pues se consiguen aumentar estos porcentajes en todas las clases y sobre todo en las minoritarias. En una red de neuronas normal, si no se

replican los patrones, se obtienen peores resultados para estos dominios, pues las clases minoritarias no llegan a ser abarcadas correctamente.

Respecto al ***metodo2***, que consiste en dividir el conjunto de entrenamiento en otros dos: uno del entrenamiento en sí de la red y otro para la validación de ésta (el *fitness* del algoritmo evolutivo); en los conjuntos de test se empeoran los resultados proporcionados por el *metodo1*. Hay que resaltar que se mejoran los aciertos con el conjunto *entrenamiento2*, el cuál es el utilizado para calcular el *fitness* para el algoritmo evolutivo, y por tanto de encauzar la evolución de éste.

En el ***metodo3***, la replicación de patrones difusos no ayuda en aquellos dominios en los que el *metodo1* no funciona (por ejemplo, *BUPA*), y fue formulado en un principio para estos casos. Se consiguen buenos resultados, pero no tanto como los conseguidos con el *metodo1*. Donde menos diferencia hay entre estos dos métodos, es en el dominio de *Balance-Scale*, y es debido a que la mayor parte de los patrones difusos son patrones mal clasificados (ver tabla de distribución de patrones difusos en el ***Apartado 4.4.4***).

Como conclusión final, de los tres métodos validados, la replicación de los patrones mal clasificados (***metodo1***) es lo que mejor resultados ha proporcionado. Es cierto que en dominio *BUPA* no se han logrado nuestros objetivos, pero se trataba de un dominio con pocos patrones. Aún así, tras conseguir buenos resultados en los otros tres dominios, podemos asegurar de que con la replicación de patrones mal clasificados se consiguen mejorar los resultados de clasificación de una red neuronal, y para aquellos que poseen datos desequilibrados, se mejoran muy satisfactoriamente.

En cuanto a trabajos futuros, en primer lugar se puede ampliar la experimentación a muchos otros dominios, para afianzar los resultados obtenidos y ver la eficacia del proyecto en dominios de otra índole. También se puede contemplar el desarrollo de otros “métodos”, como por ejemplo una combinación del *metodo1* y del *metodo3*, en el que se repliquen los patrones que sean mal clasificados o difusos. También, otra línea de trabajo puede ser la utilización de esta misma idea pero para otros sistemas de clasificación (por ejemplo con un clasificador bayesiano).

6.- PRESUPUESTO

Antes de comenzar cualquier proyecto, es conveniente dividir el trabajo a realizar en distintas tareas, para una mejor organización y seguimiento del proyecto. A continuación, se muestran las distintas fases del desarrollo del proyecto y el periodo que han durado las mismas:

Fase	Periodo		Duración (días)
	Inicio	Fin	
Análisis inicial	17-nov-08	12-dic-08	19
Diseño e implementación	15-dic-08	20-feb-09	37
Pruebas iniciales	23-feb-09	13-mar-09	15
Realización de mejoras	16-mar-09	24-abr-09	24
Experimentación	27-abr-09	19-jun-09	40
Análisis de resultados	14-sep-09	6-nov-09	39
Documentación	9-nov-09	18-jun-10	137
Correcciones finales y presentación	13-sep-10	22-oct-10	28
TOTAL	**17-nov-08**	**22-oct-10**	**339**

Para conocer con mayor precisión el tiempo empleado en la realización de las distintas fases y del proyecto en general, no se han considerado los fines de semana y no los siguientes días festivos y/o no lectivos:

- 8-dic-08: *Fiesta Nacional*
- 22-dic-08 al 7-ene-09: *Navidades*
- 6-abr-09 al 13-abr-09: *Semana Santa*
- 22-jun-09 al 13-sep-09: *Vacaciones estivales*
- 12-oct-09: *Fiesta Nacional*
- 7 y 8-dic-09: *Fiesta Nacional*
- 23-dic-09 al 7-ene-10: *Navidades*
- 19-mar-10: *Fiesta Nacional*
- 29-mar-10 al 5-abr-10: *Semana Santa*
- 3 y 4-jun-10: *Fiesta Nacional*
- 21-jun-10 al 12-sep-10: *Vacaciones estivales*
- 11 y 12-oct-10: *Fiesta Nacional*

Durante la realización del proyecto, se han utilizado una serie de recursos tanto materiales como humanos, los cuales son:

- Ingeniero Informático. El salario es de 20€/hora. La dedicación al proyecto a variado a lo largo del desarrollo del proyecto, pues a partir del 1-feb-10 ha estado trabajando a jornada completa como becario, y el tiempo disponible para la realización del presente trabajo fue menor:
    - 17-nov-08 al 31-ene-10: 220 días con una dedicación de 5horas/día
    - 1-feb-10 al 22-oct-10: 119 días con una dedicación de 1hora/día

  Por tanto el número de horas empleadas fue de:

  (220 días * 5 horas/día) + (119 días * 1 hora/día) = 1.219 horas

- Ordenador: AMD 3GHz, 4 GB RAM, Windows Vista. Con una amortización a tres años, es puede estimar un coste total de 400€.
- Máquina Linux cedida por el departamento de Inteligencia Artificial de la Universidad Carlos III de Madrid. Sin ningún coste.
- Material de oficina: folios, bolígrafos, etc. y recursos de reprografía: fotocopias, encuadernaciones, etc.

En base a los recursos utilizados y al coste de los mismos, se procede a calcular el coste total del proyecto. Además, los riesgos asociados al proyecto se han estimado en un 15% y se ha establecido un 5% de beneficios. En resumen, la siguiente tabla muestra el coste total del proyecto:

CONCEPTO	COSTE (€)
Recursos humanos	24.380’00
Hardware	400’00
Material de oficina y reprografía	85’00
Riesgo (15%)	3.729’75
Beneficios (5%)	1.429’74
Coste total	30.024’49
I.V.A. (18%)	5.404’41
Coste total con I.V.A.	**35.428’9**

El presupuesto total de este proyecto asciende a la cantidad de **35.428’90 €**, treinta y cinco mil cuatrocientos veintiocho euros con noventa céntimos de euro.

Leganés, a 22 de octubre del 2.010

Pablo L. Grande Benito

7.- BIBLIOGRAFÍA

[1] Pedro Isasi and Inés Mª Galván. *Redes de neuronas artificiales: un enfoque práctico*. Pearson Prentice Hall, 2004.

[2] D.E. Rumelhart, G.E. Hinton & R.J. Williams (1986). *Learning internal representations by error propagation*. In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Vol. 1) (pp. 318-362). Cambridge, MA: MIT Press.

[3] Charles R Darwin. *The Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*. Penguin Books Ltd., Nueva York, EE. UU., 1959. Publicado originalmente en 1859.

[4] John H. Holland. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.

[6] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991, 385 p.

[7] Christian D. von Lücken. *Algoritmos Evolutivos para Optimización Multiobjetivo: Un Estudio Comparativo en un Ambiente Paralelo Asíncrono*. Tesis (Master en Ingeniería de Sistemas). Universidad Nacional de Asunción, Paraguay, 2003.
(www.cnc.una.py/invest/paper2/lucCACIC.pdf)

[8] R. S. Rosenberg. *Simulation of genetic populations with biochemical properties*. PhD thesis, University of Michigan, Ann Arbor, Michigan, 1967.

[9] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[10] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms***.** In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pages 93-100. Lawrence Erlbaum, 1985.

[11] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and T. Meyarivan. *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II*. In IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182–197, April 2002.

[12] Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele. *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Technical Report 70. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.
(http://neo.lcc.uma.es/emoo/zitzler99b.ps.gz)

[13] Y. Jin and B. Sendhoff. *Pareto-based multiobjective machine learning: An overview and case studies*, IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 38, no. 3, pp. 397–415, May 2008.

[14] Andrzej Osyczka. *Evolutionary Algorithms for Single and Multicriteria Design Optimization, Physica*. Verlag, Germany, 2002, ISBN 3-7908-1418-0.

[15] Nitesh V. Chawla, Nathalie Japkowicz, Aleksander Kotcz. *Editorial: special issue on learning from imbalanced data sets*. SIGKDD Explorations 6(1): 1-6 (2004).
(http://www.sigkdd.org/explorations/issues/6-1-2004-06/edit_intro.pdf)

[16] Nitesh V. Chawla. *Data Mining for Imbalanced Datasets: An Overview*. The Data Mining and Knowledge Discovery Handbook 2005: 853-867.
(http://www.springerlink.com/content/r824814907175608/fulltext.pdf)